Creatio Academy

Section

Version 8.0

Table of Contents

Section

Beginner

A **section** is a UI element that represents a specific business entity and contains a set of records. For example, [*Accounts*], [*Contacts*], [*Activities*], etc. Sections are available from the side panel. You can group sections into workplaces to improve usability for particular roles. Learn more about sections in the user documentation guide: Section.

Section containers

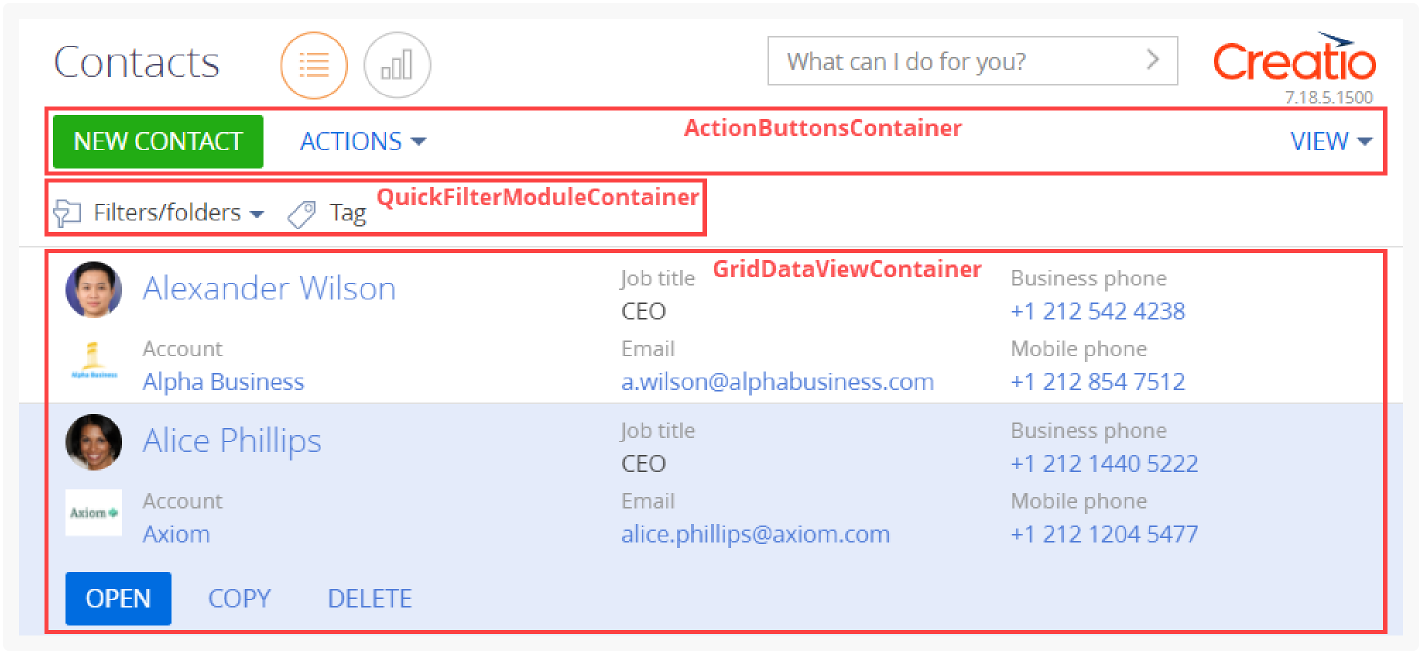
Creatio stores section UI elements in the corresponding containers. Containers are configured in the base schema of the section or replacing view model schema of the section. Containers depend on the section view.

Sections have the following **views**:

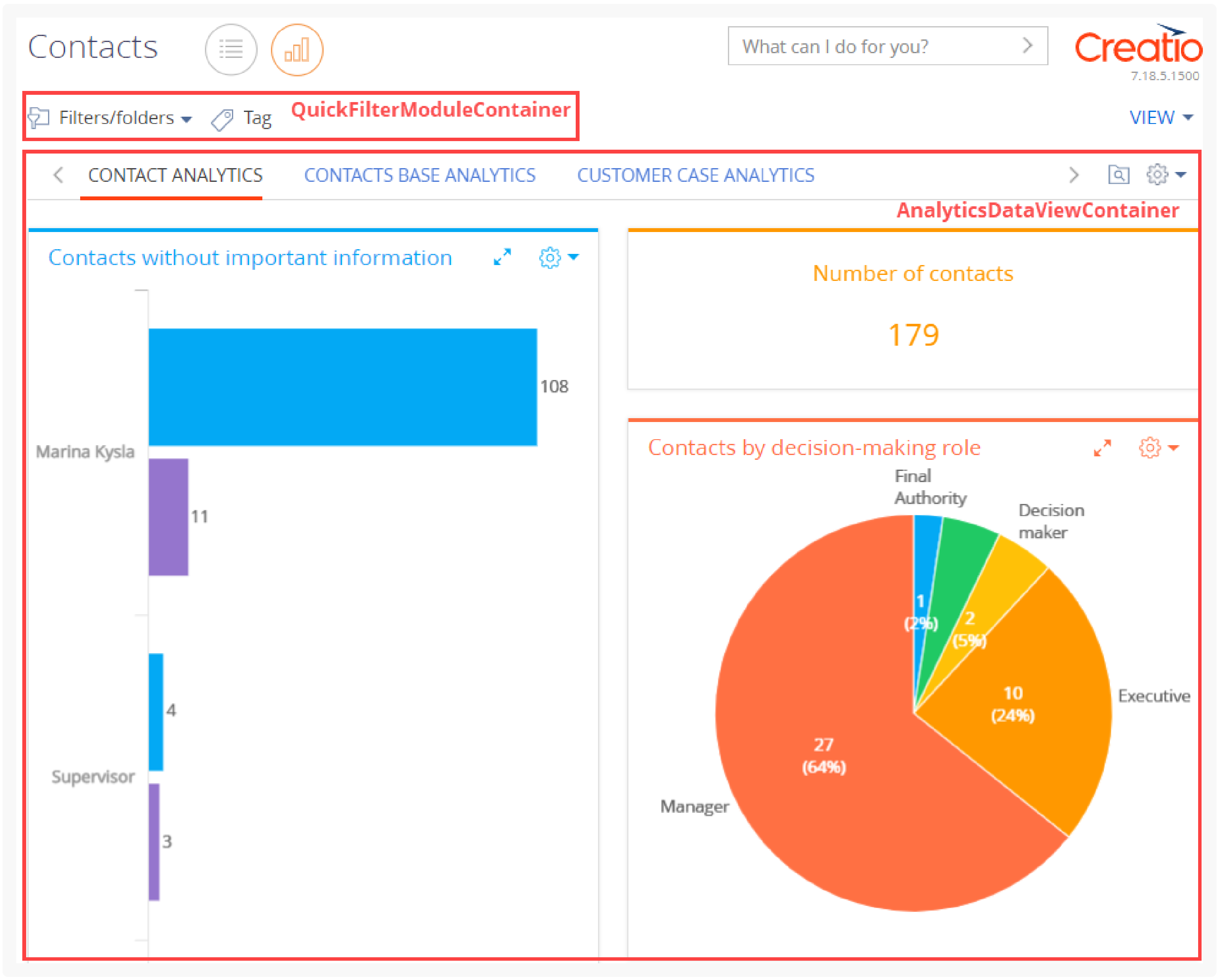
- Section list
- Section analytics

Note. Creatio uses HTML container meta names. It generates actual IDs that match the HTML elements of the section based on the meta names.

View the main section **list containers** in the figure below.



View the main section **analytics containers** in the figure below.



- The action button container (`ActionButtonsContainer`) contains the section action button and the drop-down action list button.
- The filter container (`QuickFilterViewContainer`) contains filters and tags.
- The container of the section list view (`GridDataViewContainer`) contains section records. Record edit, copy, and delete action buttons reside in the active list record. Displayed in the section list.
- The section analytics view container (`AnalyticsDataViewContainer`) contains the section dashboards. Displayed in the section analytics.

Section structure

Sections have the following **components**:

- **List**. A component that displays the section record index in tile or list view. Displayed in the `GridDataViewContainer` container.
- **Analytics**. A component that visualizes statistical data using charts, singular metrics, or lists. Dashboard blocks and custom dashboards are displayed in the `AnalyticsDataViewContainer` container.

- **Actions**. A functional element that represents a set of operations with the active section list record. Call actions using buttons located in the `ActionButtonsContainer` container and active section list record.
- **Filter**. A tool for condition-based data search and segmentation. Displayed in the `QuickFilterViewContainer` container.
- **Tag**. A label for manual record segmentation. Displayed in the `QuickFilterViewContainer`, similar to the filter.

List

A **list** is a UI element that displays the index of records added to the section or detail. For example, a contact index in the [*Contacts*] section.

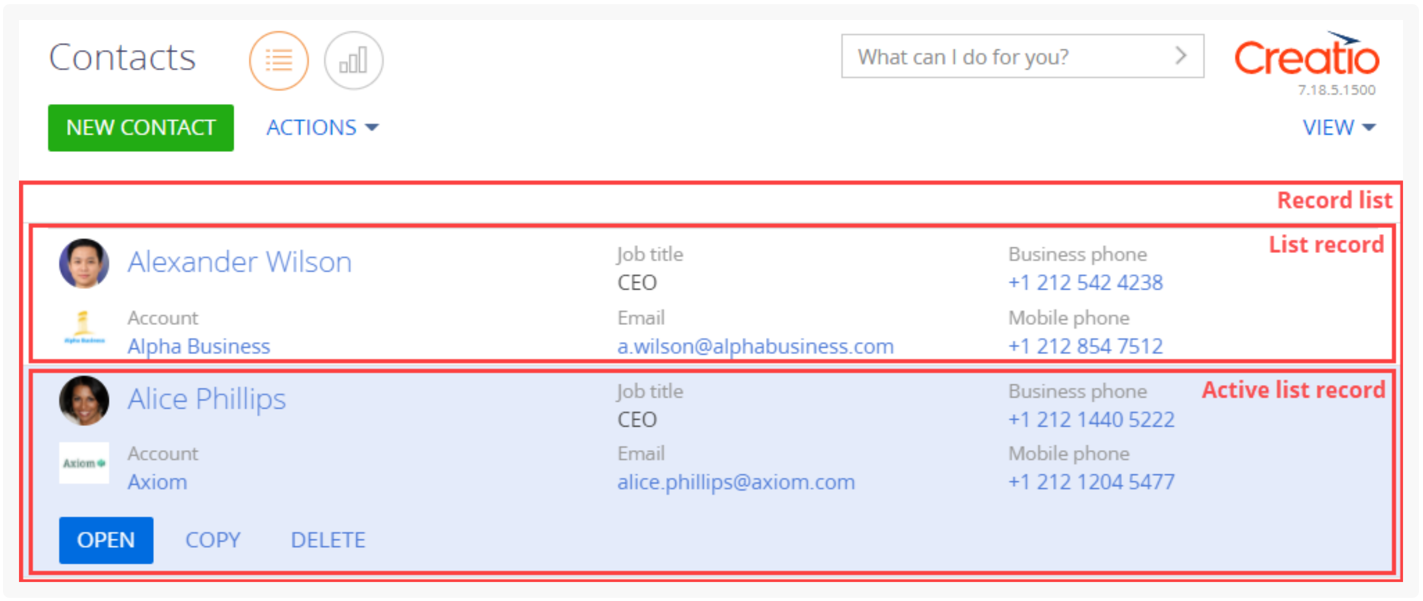
Section lists come in the following **types**:

- **Vertical list**. Enables users to switch between records without closing the page.
- **Editable list**. Enables users to edit records directly in the list without opening the pages.

A section list has the following **views**:

- **Tile view**. Displays list record fields over multiple rows. Default view.
- **List view**. Displays records as a simple table. Each record corresponds to a single row. The display order of fields can differ between the list and tile views.

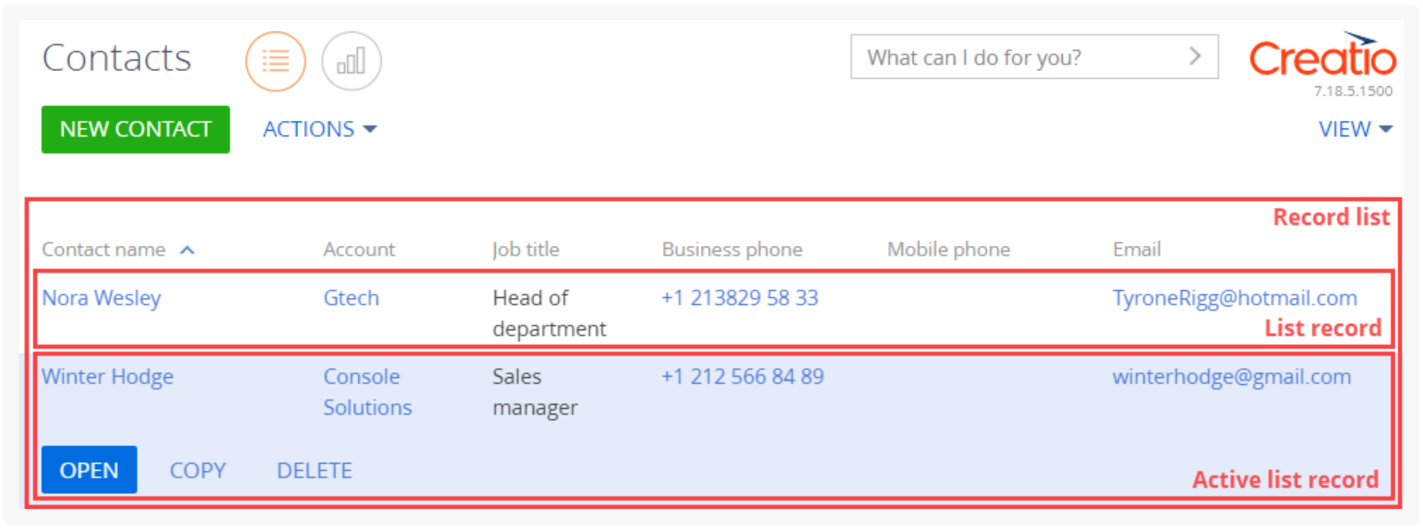
View the tile view of the [*Contacts*] section list in the figure below.



Alexander Wilson
Job title: CEO
Business phone: +1 212 542 4238
Account: Alpha Business
Email: a.wilson@alphabusiness.com
Mobile phone: +1 212 854 7512

Alice Phillips
Job title: CEO
Business phone: +1 212 1440 5222
Account: Axiom
Email: alice.phillips@axiom.com
Mobile phone: +1 212 1204 5477

View the list view of the [*Contacts*] section list in the figure below.



Contact name	Account	Job title	Business phone	Mobile phone	Email
Nora Wesley	Gtech	Head of department	+1 213829 58 33		TyroneRigg@hotmail.com
Winter Hodge	Console Solutions	Sales manager	+1 212 566 84 89		winterhodge@gmail.com

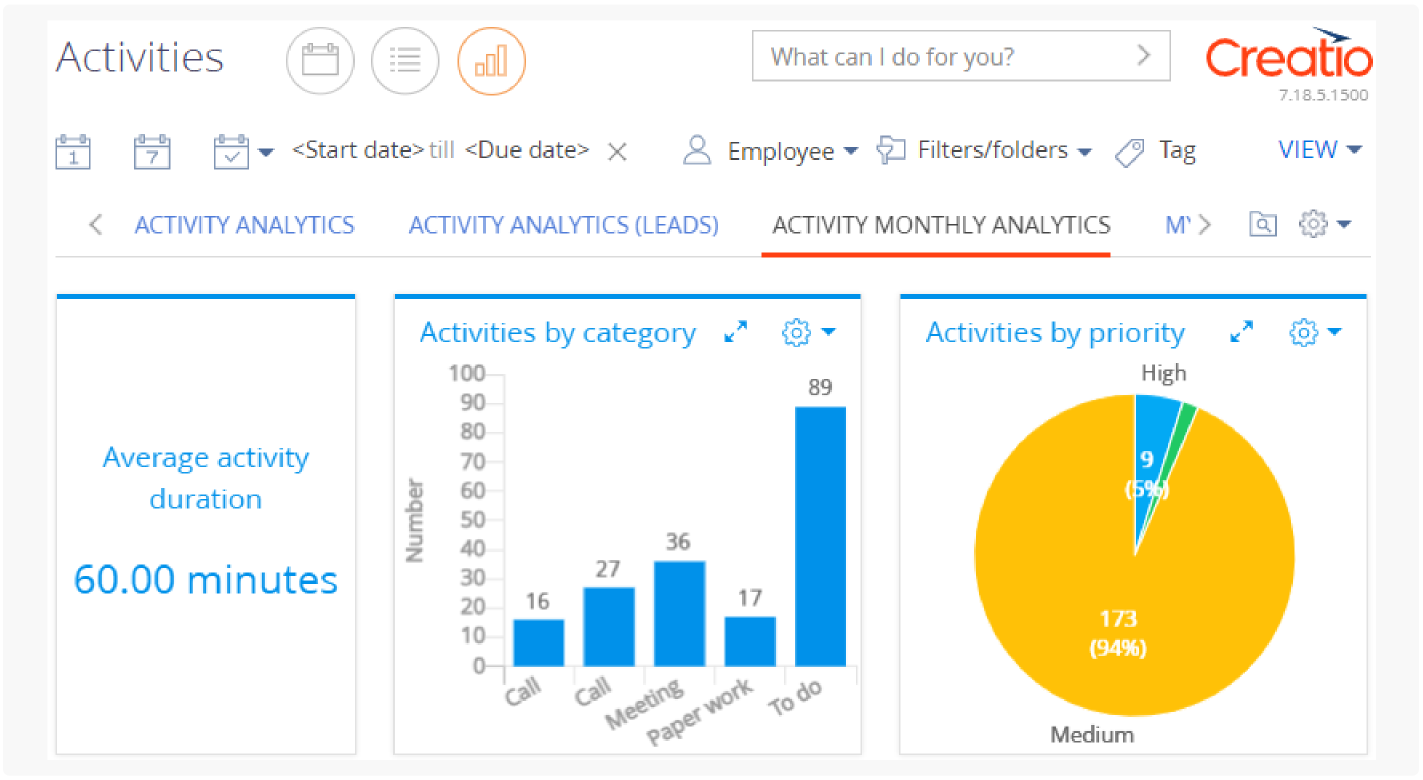
Each section corresponds to a configuration schema. The schema describes the structure of the corresponding database table with record data and contains specific instructions on processing the data. For example, the `ContactSectionV2` schema configures the [*Contacts*] section. All section schemas inherit from the `BaseSectionV2` base section schema. Each table row loosely corresponds to a section list record. For example, the [*Contacts*] section corresponds to the `Contact` business object schema that contains the index and properties of the `[Contact]` table rows. By default, the section list does not display every column of the corresponding database table. This is done to keep the displayed data easily digestible. You can view every column on the section record page.

Edit the appearance of the list, the index of displayed fields, and the sorting of data in the drop-down menu of the section toolbar's [*View*] button. Learn more in the user documentation: Work with record lists.

Analytics

The **purpose** of section analytics is to analyze the statistical data of the section. By default, Creatio displays the list section view. To **switch to the analytics section view**, click the [*Dashboards*] button next to the section name. To **open the analytics view of all sections**, open the [*Dashboards*] section.

View an example of [*Activities*] section analytics in the figure below.

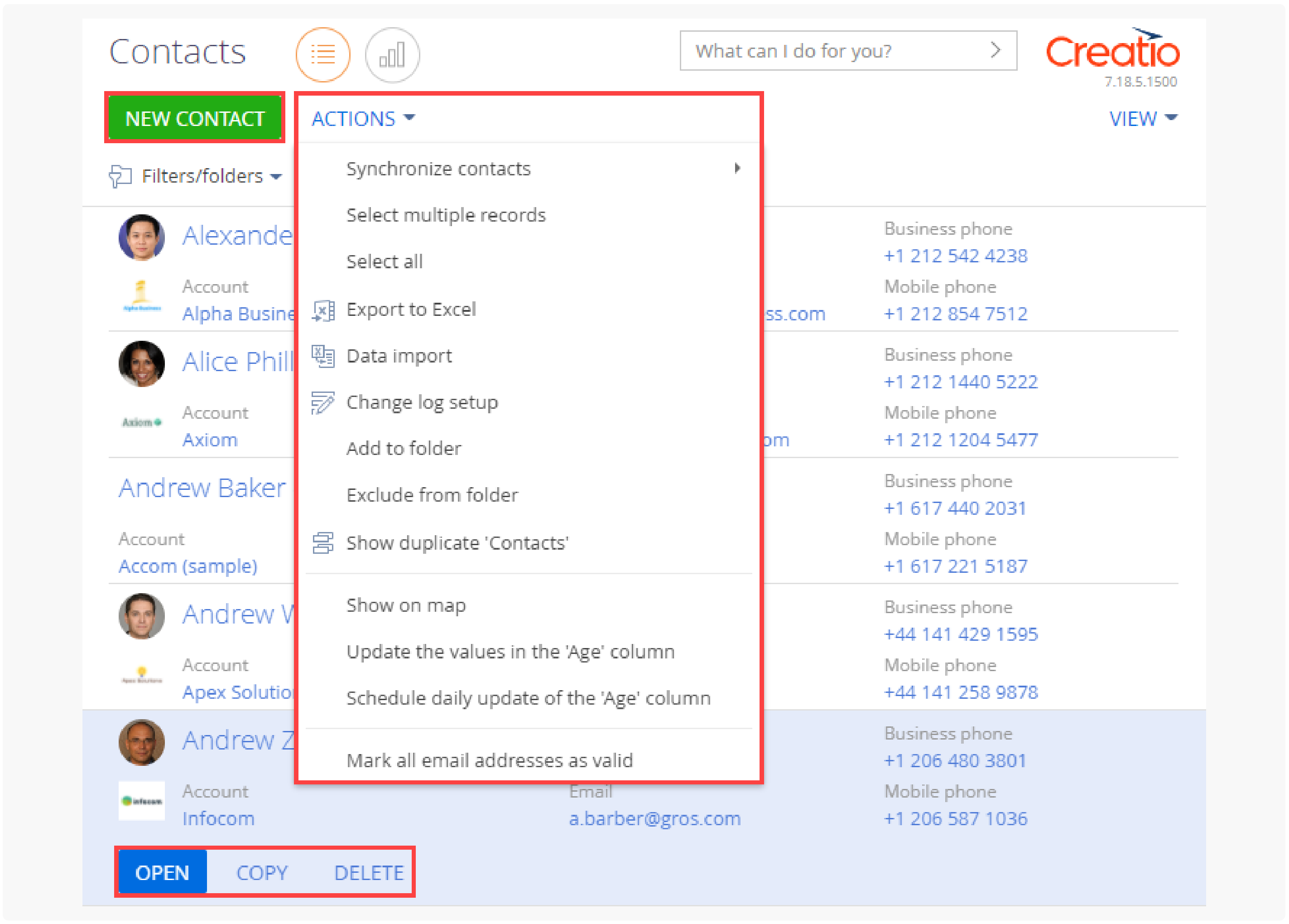


Dashboards display analytical data. Learn more about dashboards in a separate article and the user documentation guide: [Dashboard widgets](), [Analytics](). Dashboards are located on **dashboard tabs**. Learn more about dashboard tabs in the user documentation: [Add analytics in section]().

Actions

Section actions are functional elements that represent an index of operations with one or more list records. You can run actions using buttons of different types. The buttons are located both in the action container of the current section and the container of the active record.

View the [*Contacts*] section actions in the figure below.



Section actions have the following **types**:

- standard
- additional

The **standard** [*Contacts*] section actions are as follows:

- [*New contact*]. Opens a pop-up box that lets you add and save a new section record.
- [*Open*]. Opens the page of the active section record.
- [*Copy*]. Opens a section record page, copies the active record data into the page, and creates a new record when you click [*Save*].
- [*Delete*]. Deletes the active record.
- [*Select multiple records*]. Selects multiple list records.
- [*Select all*]. Selects all list records.
- [*Export to Excel*]. Exports all list records of the current section to an *.xlsx file. Learn more about data export in the user documentation: Excel data export.
- [*Data import*]. Imports data from an *xlsx file into Creatio. Learn more about data import in the user documentation: Excel data import.
- [*Change log setup*]. Opens the logging management page that also lets you select the section columns to log

when a record is edited.

Additional actions implement the functionality that depends on the section business logic. Additional [*Contacts*] section actions are as follows:

- [*Synchronize now*]. Synchronizes Creatio with Google contacts. Creatio synchronizes only records tagged as per the synchronization settings. Learn more about contact synchronization in the user documentation: Synchronize contacts and activities with Google.
- [*Add new account for synchronization*]. Synchronizes Creatio Cloud with Google. Learn more about synchronization in the user documentation: Synchronize contacts and activities with Google.
- [*Add to folder*]. Opens a pop-up box that lets you select a folder where to place the active record.
- [*Exclude from folder*]. Excludes the active record from all of its current folders.
- [*Show duplicate 'Contacts'*]. Opens an additional page that contains all likely duplicates of the contacts. Creatio adds records to this page automatically after performing the duplicate search.
- [*Show on map*]. Lets you display the location of selected contacts on the map. The action opens a box with a map that marks the contacts selected in the list. If the address fields of the selected contacts are empty, Creatio does not perform the action. If the address fields of some contacts are empty or filled out incorrectly, the box displays the corresponding information.
- [*Update the values in the 'Age' column*]. Updates the [*Age*] column on the active contact page.
- [*Schedule daily update of the 'Age' column*]. Opens a pop-up box that lets you set up the time of the daily contact age update.
- [*Mark all email addresses as valid*]. Selects the [*Valid*] checkbox for the email address of the active contact record.

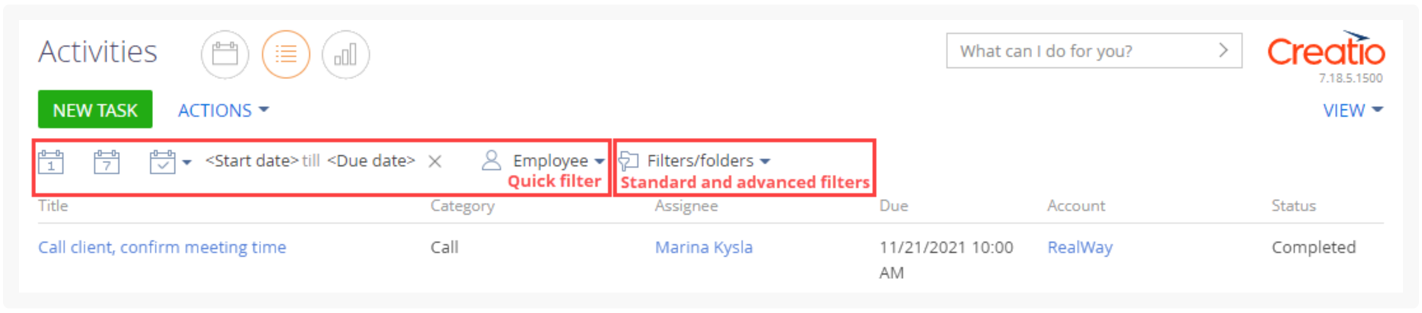
Filters

A **filter** is a tool for condition-based data search and segmentation. You can use filters as an individual tool in Creatio sections. Filters are also available as part of other tools, for example, when you set up dynamic folders, dashboards, or business processes. Creatio displays filter controls above the section list. Learn more about filters in the user documentation: Filters.

Filters have the following **types**:

- quick filter
- advanced filter
- standard filter

Configure the quick filter on the toolbar. Configure the standard and advanced filters in the [*Filters/folders*] menu.

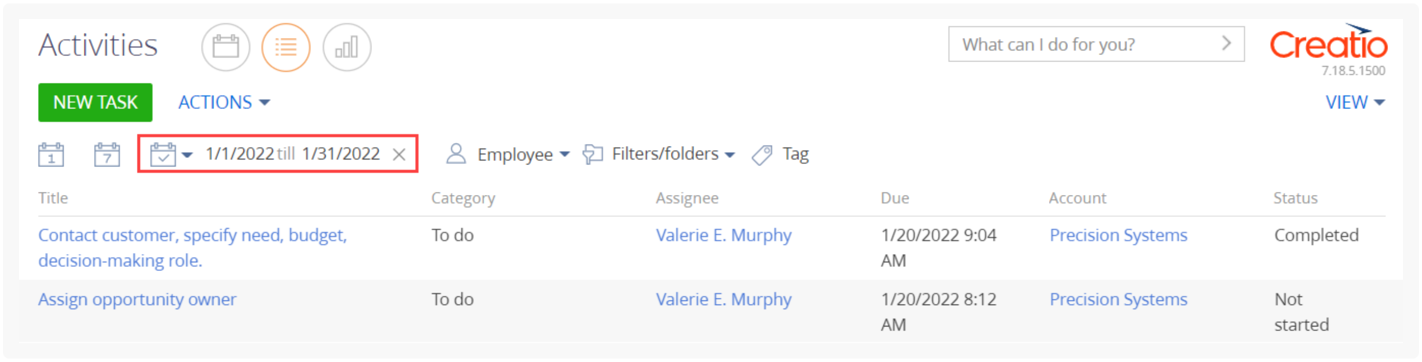


Title	Category	Assignee	Due	Account	Status
Call client, confirm meeting time	Call	Marina Kysla	11/21/2021 10:00 AM	RealWay	Completed

Quick filter is available in certain Creatio sections. Use quick filters to filter data by the most frequently used parameters (typically, by the date and owner). For example, the [*Activities*] section includes a quick filter since you most often need to view the activity of a single employee during the specified period. Learn more about the quick filter in the user documentation: Filters.

To **set the quick filter**, select the period to filter the section list records in the [*Select period*] drop-down menu.

View an example of a quick filter that filters activities by current month (the [*Select period*] drop-down menu is set to [*Current month*]) in the figure below.



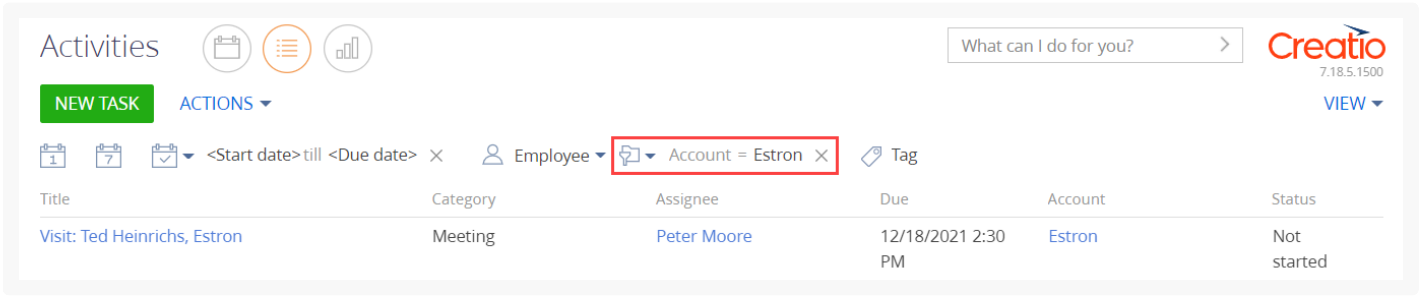
Title	Category	Assignee	Due	Account	Status
Contact customer, specify need, budget, decision-making role.	To do	Valerie E. Murphy	1/20/2022 9:04 AM	Precision Systems	Completed
Assign opportunity owner	To do	Valerie E. Murphy	1/20/2022 8:12 AM	Precision Systems	Not started

Standard filter lets you search for records by the values of one or more current section columns quickly. Available in most section lists. Learn more about the standard filter in the user documentation: Filters.

To **set the standard filter**:

- Select [*Add filter*] in the [*Filters/folders*] drop-down menu.
- Select the field and field value to filter the section list records.

View an example of a standard filter that filters activities by account (the [*Account*] field is set to [*Estron*]) in the figure below.



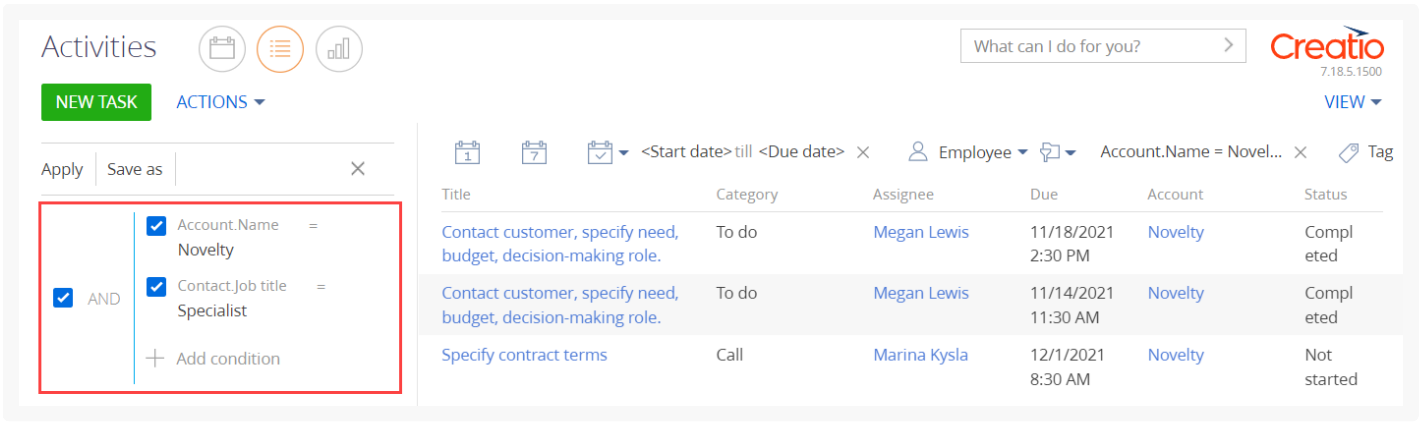
Title	Category	Assignee	Due	Account	Status
Visit: Ted Heinrichs, Estron	Meeting	Peter Moore	12/18/2021 2:30 PM	Estron	Not started

Advanced filter consists of several parameters and complex search conditions. For example, you can use the advanced filter to display meetings with new customers in the [*Activities*] section. You can also save the parameters and conditions of an advanced filter as a dynamic folder to reuse them later. Learn more about the advanced filter in the user documentation: Filters.

To **set the advanced filter**:

- Select [*Switch to advanced mode*] in the [*Filters/folders*] drop-down menu.
- Select the object, object column, and column value to filter the section list records.

View an example of an advanced filter that filters activities by account (the [*Name*] field of the [*Account*] object is set to [*Novelty*]) and job title (the [*Job title*] field of the [*Contact*] object is set to [*Specialist*]) in the figure below.



Account.Name = Novelty

Contact.Job title = Specialist

Title	Category	Assignee	Due	Account	Status
Contact customer, specify need, budget, decision-making role.	To do	Megan Lewis	11/18/2021 2:30 PM	Novelty	Completed
Contact customer, specify need, budget, decision-making role.	To do	Megan Lewis	11/14/2021 11:30 AM	Novelty	Completed
Specify contract terms	Call	Marina Kysla	12/1/2021 8:30 AM	Novelty	Not started

A **folder** is a filtering tool that groups records by specific conditions. Folders can have hierarchical structure. Learn more about folders in the user documentation: Folders.

Folders have the following **types**:

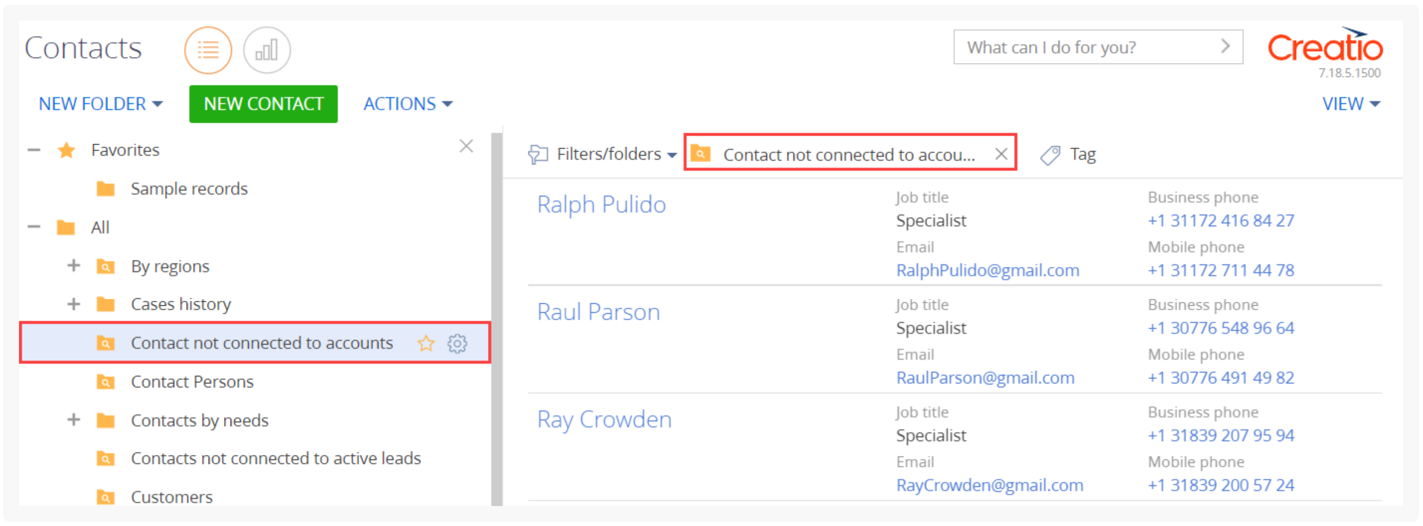
- **Dynamic folders**. Contain only the section records that match the specified filter conditions. For example, you can filter records by creation date to create a [*New customers*] dynamic folder. Creatio generates and updates the content of such folders automatically.
- **Static folders**. Contain only the section records added by users, manually or as part of the conversion from a dynamic folder. For example, you can create a [*VIP customers*] static folder to ensure sales associates or their managers make personal decisions to add customers to the folder.

You can only add records to or remove them from static folders. Creatio adds a record to a folder automatically if the record meets the filter conditions of the folder. If a record no longer meets the filter conditions of the folder, Creatio removes the record from the folder automatically.

To **set up a folder**:

- Select [*Show folders*] in the [*Filters/folders*] drop-down menu.
- Select the folder to filter the section list records.

View an example folder that includes contacts not connected to accounts (the [*Contacts not connected to accounts*] folder) in the figure below.



Ralph Pulido	Job title: Specialist Email: RalphPulido@gmail.com	Business phone: +1 31172 416 84 27 Mobile phone: +1 31172 711 44 78
Raul Parson	Job title: Specialist Email: RaulParson@gmail.com	Business phone: +1 30776 548 96 64 Mobile phone: +1 30776 491 49 82
Ray Crowden	Job title: Specialist Email: RayCrowden@gmail.com	Business phone: +1 31839 207 95 94 Mobile phone: +1 31839 200 57 24

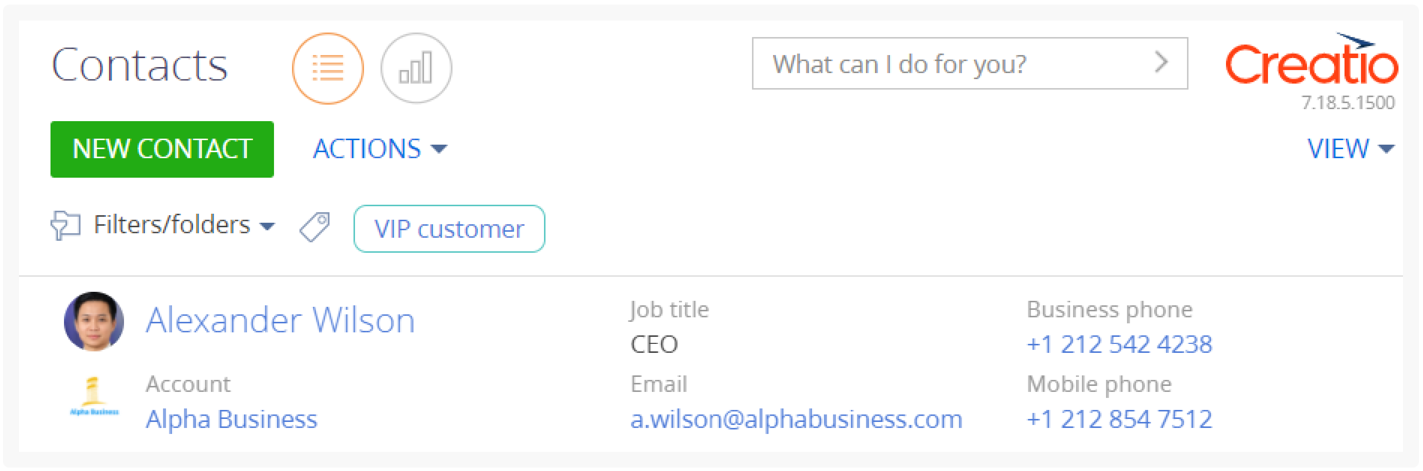
Tags

A **tag** is a special label you can use to segment records manually. For example, you can tag records in the [*Contacts*] section to mark VIP customers or ban customers. Tag records manually. To **select a tag**, click [*Tag*] and select a tag to filter the section list records. Learn more about tags in the user documentation: Tags.

Creatio provides the following tag **types**:

- **Private** tags. Only visible and accessible to the user who created them. Any user can create any number of private tags. Private tags are not visible to managers and system administrators. Creatio marks private tags green.
- **Corporate** tags. Visible to the company employees. Any employee can add or remove a corporate tag. Any employee/role that has permissions for the [*Corporate tags management*] (`CanManageCorporateTags` code) operation can create corporate tags. Creatio marks corporate tags blue.
- **Public** tags. Visible to the company employees, as well as self-service portal users. Any employee can add or remove a public tag. Any employee/role that has permissions for the [*Public tags management*] (`CanManagePublicTags` code) operation can create public tags. Creatio marks public tags red.

View an example of contacts that are VIP customers ([*VIP customer*] tag selected) in the figure below.



Alexander Wilson Account: Alpha Business	Job title: CEO Email: a.wilson@alphabusiness.com	Business phone: +1 212 542 4238 Mobile phone: +1 212 854 7512

Create a section

You can create a section in the Section Wizard. To create a section, follow the instructions in the user documentation: Create a new section.

View object schemas and client schemas that the Section Wizard creates in the custom package when you add a section in the table below.

Schemas that the Section Wizard creates

Name	Purpose	Parent object
	Object schemas	
`[Section object name]`	The primary section object	`BaseEntity` base object.
`[Section object name]Folder`	The object folder. A utility object that ensures the correct grouping of section records. Structures the section folder tree.	`BaseFolder` base folder.
`[Section object name]InFolder`	The object in the folder. A utility object that ensures grouping section records operates as intended. Defines links between section records and their folders.	`BaseItemInFolder` base element in the folder.
`[Section object name]File`	The object for the [*Attachments*] detail.	`File` file.
`[Section object name]Tag`	The section tag.	`BaseTag` base tag.
`[Section object name]InTag`	The tag in the section object.	`BaseEntityInTag` base tag in the base object.
	Client schemas	
`[Section object name]Section`	The section schema.	`BaseSectionV2` base section schema.
`[Section object name]Page`	The schema of the section record page.	`BaseModulePageV2` base schema of the section record page.

Add a custom column to the section list

To add a custom column to the section list, add a column to the schema of the section business object. When you add a custom column, Creatio adds a replacing view model schema to the current package. The schema inherits the base object columns. The custom column is added to the schema.

You can use the following **tools** to add a custom column to the schema of the section business object:

- Section Wizard
- Creatio IDE

To add a custom column to the section list in the **Section Wizard**, follow the instructions in the user documentation: Work with record lists.

To add a custom column to the section page in **Creatio IDE**:

1. Create a replacing view model schema of the section page to place the custom column. To do this, follow the guide in a separate article: Client module.
2. Select the view model schema to replace as the parent object.
3. Add a custom column of the corresponding type to the replacing view model schema of the section page.
4. Set up the appearance of the custom column in the section list.

Set up the appearance conditions of the section list

Creatio lets you set up the appearance conditions of the section list. Record lists that match the specified conditions are displayed in line with the configured visual styles.

To **set up the appearance conditions of the section list**:

1. Create a replacing view model schema of the section page. To do this, follow the guide in a separate article: Client module.
2. Select the view model schema to replace as the parent object.
3. Set up the **appearance conditions of the section list**. To do this, implement the `prepareResponseCollectionItem()` method in the `methods` property. The method overloads the base method and modifies the data row before uploading the row to the list. Implement the assignment of a specific value to the `customStyle` property in the method. The property controls the list row appearance. The property is an object that has child properties that are similar to CSS properties and generate the visual style of the relevant list records.

   View an example of the `customStyle` property setup below.

   Example of the `customStyle` property setup

   ```
   item.customStyle = {
    /* Set the text color to white. */
    "color": "white",
    /* Set the background color to orange. */
    "background": "orange"
   };
   ```

Add a section record action

Creatio lets you add custom actions for either a single record or multiple records. The `BaseSectionV2` schema of the `NUI` package implements the base content of the [*Actions*] drop-down menu on the section page.

To **add an action for a single section record**:

1. Create a replacing view model schema of the section page. To do this, follow the guide in a separate article: Client module.
2. Select the view model schema to replace as the parent object.
3. The section action index is an instance of the `Terrasoft.BaseViewModelCollection` class. Each item of the action index is a view model. Set up the action in the configuration object that lets you both set the properties of the action view model explicitly and use the base binding mechanism.

   Implement the **menu item behavior**. To do this, implement the `getSectionActions()` overloaded parent schema method in the `methods` property. The method returns the collection of section actions. When you replace base sections in the `getSectionActions()` method of the replacing module, call the parent implementation of the method to initialize the actions of the parent section. To do this, execute the `this.callParent(arguments)` method that returns the action collection of the base section.

   To add an action to the collection, call the `addItem()` method. The `getButtonMenuItem()` callback method is passed to the method as a parameter. The method creates an instance of the action view model by the configuration object passed to the method as the parameter.

   Reference the selected record via the `ActiveRow` attribute of the section view model. The attribute returns the primary column value of the selected record. In turn, you can use the primary column value of the selected record to get the values uploaded to the field list of the selected object.

   The `SelectedRows` property of the section view model stores the primary column values of the selected records. You can use these values to get the values uploaded to the field list of selected objects. For example, from a regular list's data collection stored in the `GridData` view model property of the list.

   View the base implementation of action addition below.

   BaseSectionV2

```
/**
* Returns the collection of section actions in the list view mode.
* @protected
* @virtual
* @return {Terrasoft.BaseViewModelCollection} The section action collection.
*/
getSectionActions: function() {
 /* The action collection. An instance of Terrasoft.BaseViewModelCollection. */
 var actionMenuItems = this.Ext.create("Terrasoft.BaseViewModelCollection");
 /* Add an action to the collection. Pass the method that instantiates the action model insta
 actionMenuItems.addItem(
  this.getButtonMenuItem({
  /* The configuration object that sets up the action. */
```

```
      ...
      })
    );
    return actionMenuItems;
  }
```

View the properties of the section action's configuration object that is passed as a parameter to the `getButtonMenuItem()` method in the table below.

Configuration object properties

Property	Description
`Type`	The type of the [*Actions*] drop-down menu item. The property lets you add a horizontal line that separates menu blocks to the action menu. To do this, specify `Terrasoft.MenuSeparator` as the property value. If you do not specify the property value, Creatio adds a menu item by default.
`Caption`	The title of the [*Actions*] menu item. We recommend using localizable strings for titles.
`Click`	Binds an action handler method by the method name.
`Enabled`	The boolean property that controls the menu item availability.
`Visible`	The boolean property that controls the menu item visibility.

Set up the quick filters block of the section

Creatio lets you implement a custom quick filters block for section list records.

To **set up the quick filters block of the section**:

1. Create a replacing view model schema of the section page. To do this, follow the guide in a separate article: Client module.
2. Select the view model schema to replace as the parent object.
3. Implement the **filtering behavior**. To do this, implement the `initFixedFiltersConfig()` method in the `methods` property. The method initializes the fixed filters. In the method, create a configuration object that has an array of filters. Assign the object link to the `fixedFiltersConfig` attribute of the view model.

Delete a section

1. Make sure you have access to Creatio configuration and database.
2. Unlock the file that corresponds to the relevant section in the SVN repository.
3. Delete the section that is based on the relevant object. To do this, execute an SQL query.

SQL query

MSSQL

```
DECLARE @UId UNIQUEIDENTIFIER
DECLARE @ModuleEntityUID UNIQUEIDENTIFIER;
DECLARE @ModuleID UNIQUEIDENTIFIER;
DECLARE @Name NVARCHAR(max) = 'ToDelete';
select @UId = UId from SysSchema where Name Like @Name
select @ModuleEntityUID = Id from SysModuleEntity where
SysEntitySchemaUId = @UId
select @ModuleID = Id from SysModule where SysModuleEntityId = @ModuleEntityUID;
delete from SysModuleInWorkplace where SysModuleId = @ModuleID;
delete from SysModule where Id = @ModuleID;
delete from SysModuleEdit where SysModuleEntityId = @ModuleEntityUID;
delete from SysModuleEntity where Id = @ModuleEntityUID;
delete from SysDetail where EntitySchemaUId = @UId;
delete from SysLookup where SysEntitySchemaUId = @UId;
delete from [Lookup] where SysEntitySchemaUId = @UId;
```

PostgreSQL

```
do $$
DECLARE
name varchar(100) = 'UsrProperty';
moduleID UUID;
moduleEntityUID UUID;
uId UUID;
BEGIN
uId = (select "UId" from "SysSchema" where "Name" Like name);
moduleEntityUID = (select "Id" from "SysModuleEntity" where
"SysEntitySchemaUId" = uId);
moduleID = (select "Id" from "SysModule" where "SysModuleEntityId" = moduleEntityUID);
update "SysModule" set "Image32Id"=Null where "Id" = moduleID;
delete from "SysModuleInWorkplace" where "SysModuleId" = moduleID;
delete from "SysModule" where "Id" = moduleID;
delete from "SysModuleEdit" where "SysModuleEntityId" = moduleEntityUID;
delete from "SysModuleEntity" where "Id" = moduleEntityUID;
delete from "SysDetail" where "EntitySchemaUId" = uId;
delete from "SysLookup" where "SysEntitySchemaUId" = uId;
delete from "Lookup" where "SysEntitySchemaUId" = uId;
end $$;
```

Replace `ToDelete` with the name of the custom section schema.

4. After you clear the database, delete the custom schemas from the [*Configuration*] section.

Delete the configuration schemas in this **order**:

a. `ToDeleteFile`.
b. `ToDeleteInFolder`.
c. `ToDeleteInTag`.
d. `ToDeleteTag`.
e. `ToDeleteFolder`.
f. `ToDelete`.

Add an action for a single record to the section

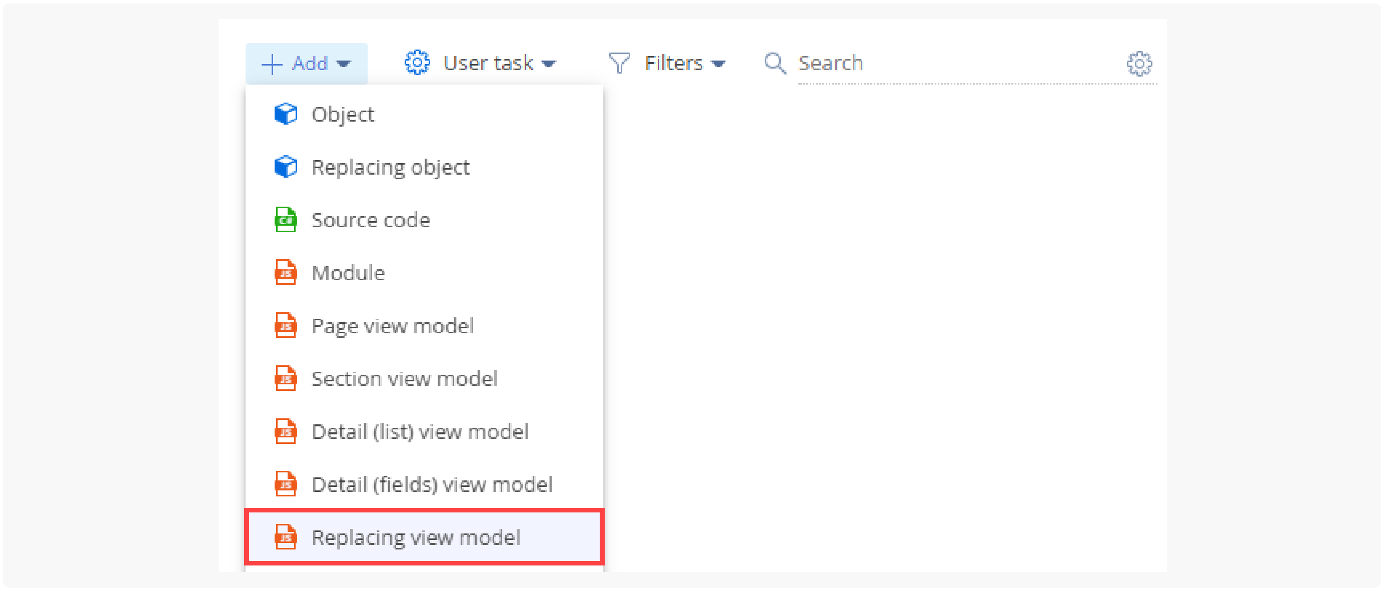
Medium

The example is relevant to Sales Creatio products.

Example. Add an action for the record of the [*Orders*] section list. The action must display the order creation date in a message box. The action must be active for orders at the [*In progress*] stage.

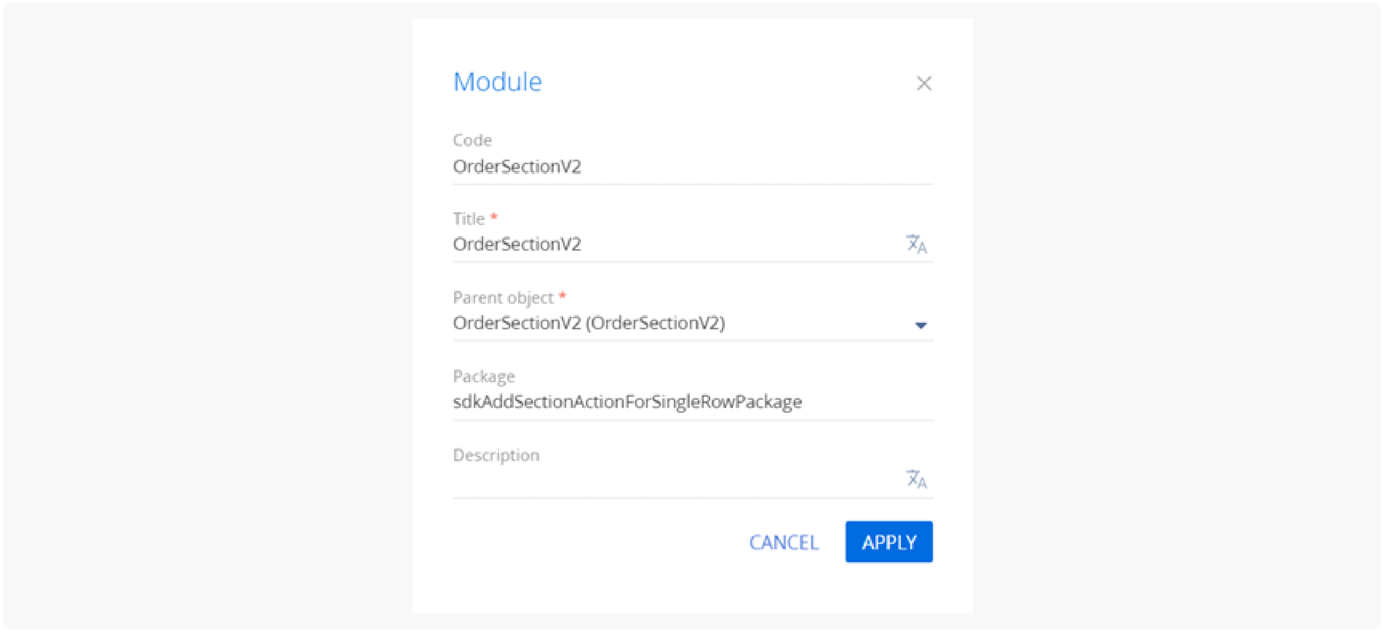
Create a schema of the replacing section view model

1. Go to the [*Configuration*] section and select a custom package to add the schema.
2. Click [*Add*] → [*Replacing view model*] on the section list toolbar.



3. Fill out the **schema properties**.

- Set [*Code*] to "`OrderSectionV2`".
- Set [*Title*] to "`Order section`".
- Set [*Parent object*] to "`OrderSectionV2`".


Module

Code: OrderSectionV2

Title *: OrderSectionV2

Parent object *: OrderSectionV2 (OrderSectionV2)

Package: sdkAddSectionActionForSingleRowPackage

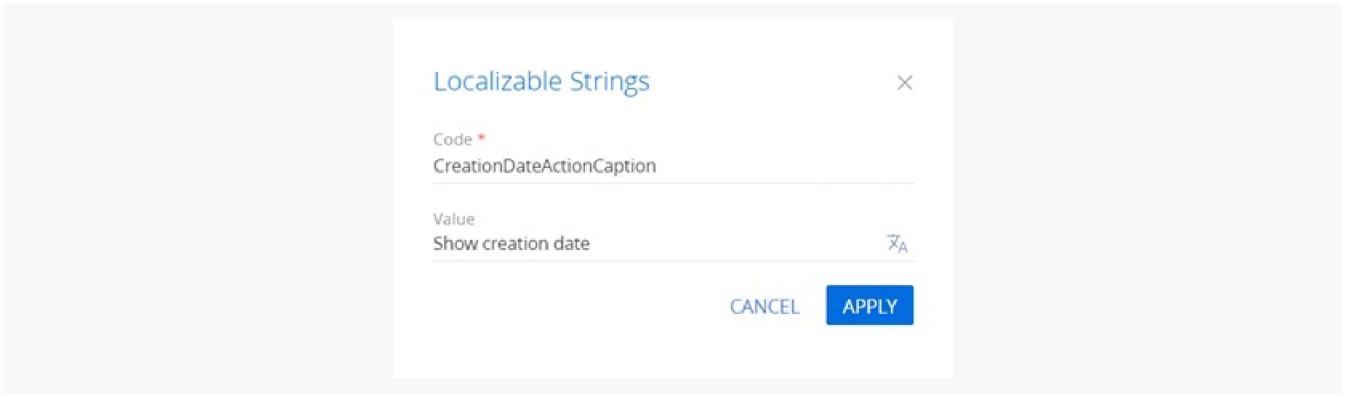
Description:

4. Add a **localizable string** that contains the menu item caption.

   a. Click the + button in the context menu of the [*Localizable strings*] node.

   b. Fill out the **localizable string properties**.

   - Set [*Code*] to "`CreationDateActionCaption`".
   - Set [*Value*] to "`Show creation date`".


Localizable Strings

Code *: CreationDateActionCaption

Value: Show creation date

   e. Click [*Add*] to add a localizable string.

5. Implement the **menu item behavior**. To do this, implement the following **methods** in the `methods` property:

- `isRunning()`. Checks whether the order selected in the list is at the [*In progress*] stage.
- `isCustomActionEnabled()`. Determines whether the menu item is available.
- `showOrderInfo()`. The action handler method that displays the order creation date in a message box.

- `getSectionActions()`. The overloaded parent schema method that retrieves the section action collection.

Reference the selected record via the `ActiveRow` attribute of the section view model. The attribute returns the primary column value of the selected record. In turn, you can use the primary column value of the selected record to get the values uploaded to the field list of the selected object.

View the source code of the replacing view model schema of the section below.

OrderSectionV2

```
define("OrderSectionV2", ["OrderConfigurationConstants"], function(OrderConfigurationConstant
    return {
        /* The name of the section object schema. */
        entitySchemaName: "Order",
        /* The methods of the section view model. */
        methods: {
            /* Check the order stage. activeRowId is the primary column value of the selected
            isRunning: function(activeRowId) {
                /* Retrieve the data collection of the section list's regular list view. */
                var gridData = this.get("GridData");
                /* Retrieve the order model by the specified primary column value. */
                var selectedOrder = gridData.get(activeRowId);
                /* Retrieve the properties of the order status model. */
                var selectedOrderStatus = selectedOrder.get("Status");
                /* Return true if the order status is [In progress], return false otherwise.
                return selectedOrderStatus.value === OrderConfigurationConstants.Order.OrderS
            },
            /* Determine whether the menu item is available. */
            isCustomActionEnabled: function() {
                /* Attempt to retrieve the ID of the active record. */
                var activeRowId = this.get("ActiveRow");
                /* Return true if the ID is defined and the order status is [In progress], re
                return activeRowId ? this.isRunning(activeRowId) : false;
            },
            /* The action handler method. Displays the order creation date in the message box
            showOrderInfo: function() {
                var activeRowId = this.get("ActiveRow");
                var gridData = this.get("GridData");
                /* Retrieve the order creation date. The column must be added to the list. */
                var dueDate = gridData.get(activeRowId).get("Date");
                /* Display the message box. */
                this.showInformationDialog(dueDate);
            },
            /* Overload the base virtual method that returns the section action collection. *
            getSectionActions: function() {
                /* Call the parent method implementation that retrieves the collection of the
                var actionMenuItems = this.callParent(arguments);
                /* Add a separator line. */
                actionMenuItems.addItem(this.getButtonMenuItem({
                    Type: "Terrasoft.MenuSeparator",
                    Caption: ""
                }));
                /* Add a menu item to the section action index. */
                actionMenuItems.addItem(this.getButtonMenuItem({
                    /* Bind the menu item caption to the localizable schema string. */
                    "Caption": {bindTo: "Resources.Strings.CreationDateActionCaption"},
                    /* Bind the action handler method. */
                    "Click": {bindTo: "showOrderInfo"},
                    /* Bind the property of the menu item availability to the value returned
                    "Enabled": {bindTo: "isCustomActionEnabled"}
                }));
                /* Return the expanded section action collection. */
                return actionMenuItems;
            }
        }
    };
});
```

6. Click [*Save*] on the Designer's toolbar.

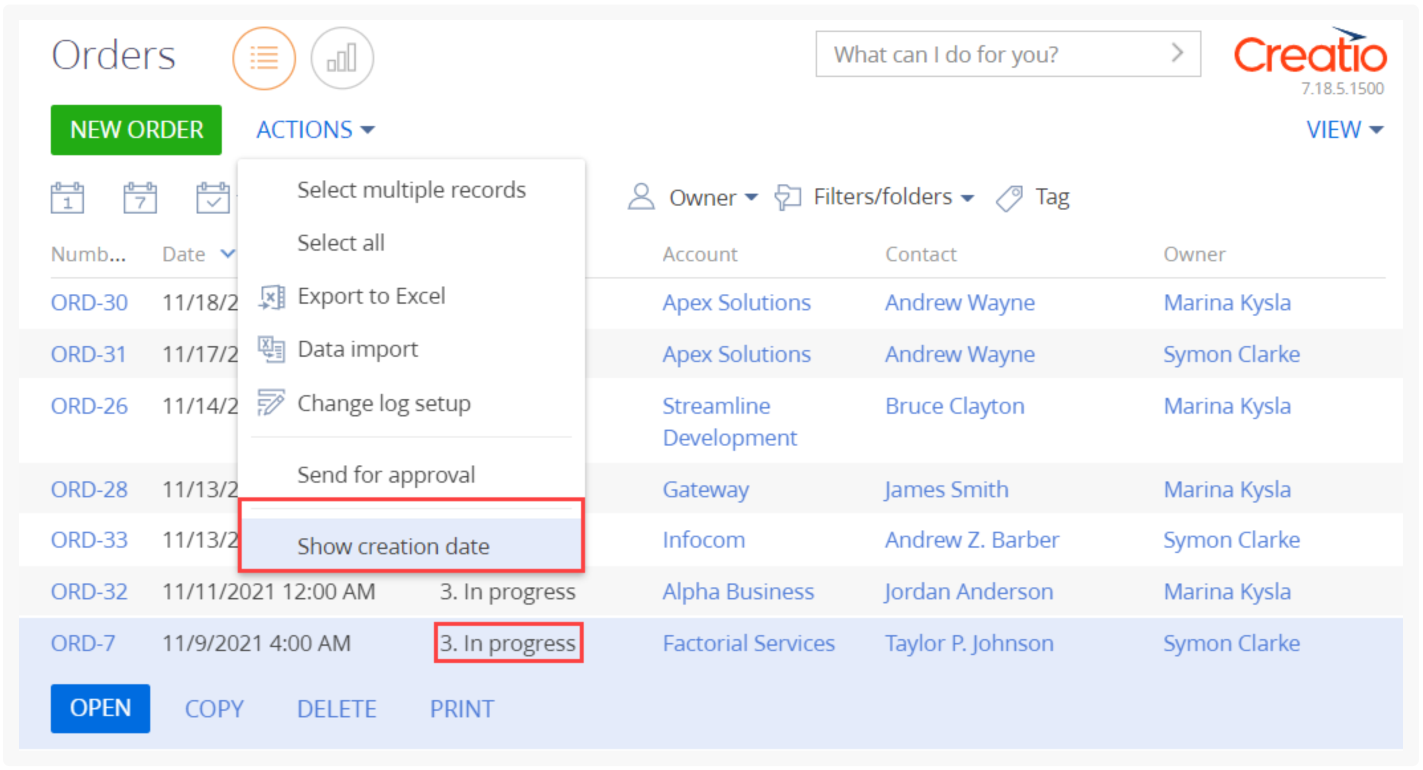
The outcome of the example

To **view the outcome of the example**:

1. Clear the browser cache.
2. Refresh the [*Orders*] section page.

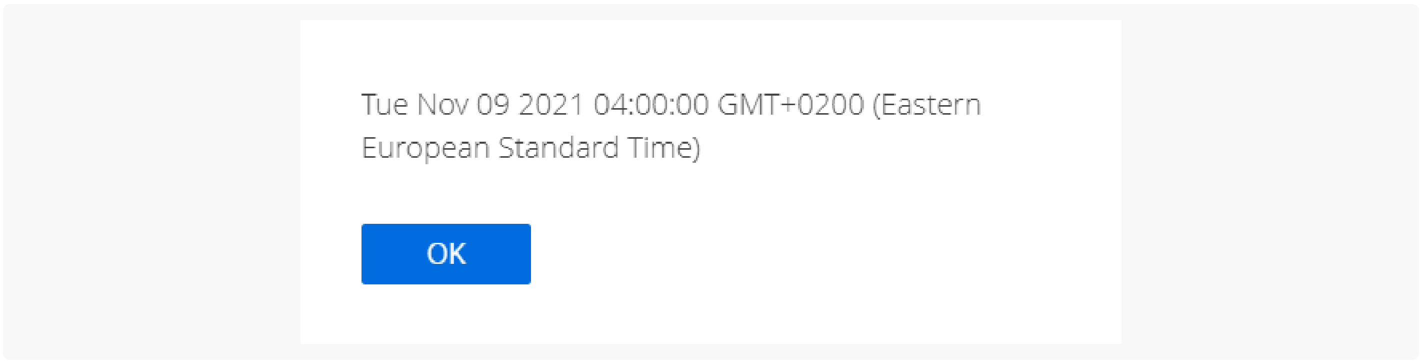
As a result, Creation will add the [*Show creation date*] action to the order page.

If the order is at the [*In progress*] stage, the [*Show creation date*] action will be active.

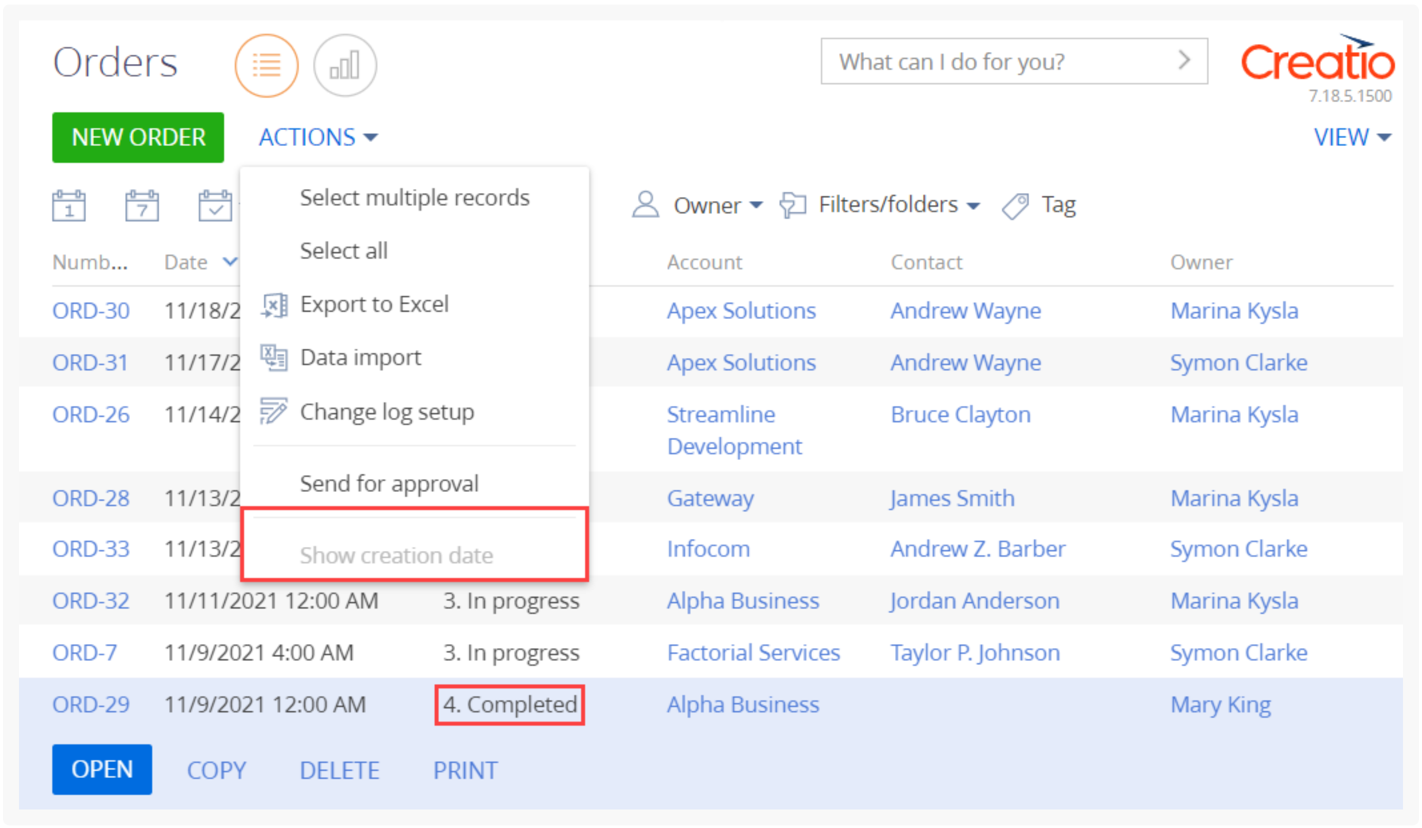


Numb...	Date		Account	Contact	Owner
ORD-30	11/18/2		Apex Solutions	Andrew Wayne	Marina Kysla
ORD-31	11/17/2		Apex Solutions	Andrew Wayne	Symon Clarke
ORD-26	11/14/2		Streamline Development	Bruce Clayton	Marina Kysla
ORD-28	11/13/2		Gateway	James Smith	Marina Kysla
ORD-33	11/13/2		Infocom	Andrew Z. Barber	Symon Clarke
ORD-32	11/11/2021 12:00 AM	3. In progress	Alpha Business	Jordan Anderson	Marina Kysla
ORD-7	11/9/2021 4:00 AM	3. In progress	Factorial Services	Taylor P. Johnson	Symon Clarke

Run the [*Show creation date*] action to bring up the message box that displays the order creation date.


Tue Nov 09 2021 04:00:00 GMT+0200 (Eastern European Standard Time)

If the order's current stage is not [*In progress*], the [*Show creation date*] action will be inactive.



Numb...	Date		Account	Contact	Owner
ORD-30	11/18/2		Apex Solutions	Andrew Wayne	Marina Kysla
ORD-31	11/17/2		Apex Solutions	Andrew Wayne	Symon Clarke
ORD-26	11/14/2		Streamline Development	Bruce Clayton	Marina Kysla
ORD-28	11/13/2		Gateway	James Smith	Marina Kysla
ORD-33	11/13/2		Infocom	Andrew Z. Barber	Symon Clarke
ORD-32	11/11/2021 12:00 AM	3. In progress	Alpha Business	Jordan Anderson	Marina Kysla
ORD-7	11/9/2021 4:00 AM	3. In progress	Factorial Services	Taylor P. Johnson	Symon Clarke
ORD-29	11/9/2021 12:00 AM	4. Completed	Alpha Business		Mary King

Add an action for multiple records to the section

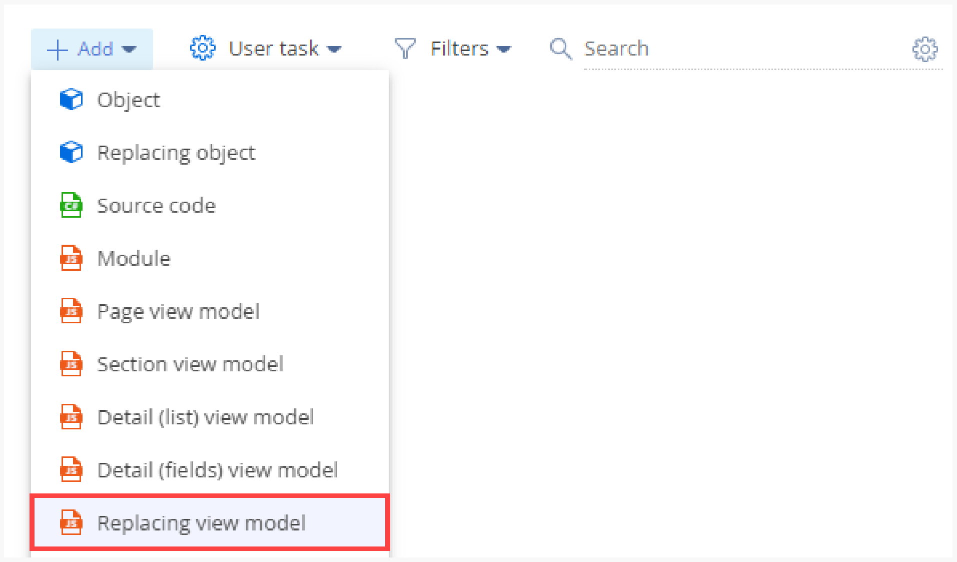
Medium

The example is relevant to Sales Creatio products.

Example. Add an action for records of the [*Orders*] section list. The action must display the index of accounts for the selected orders in the message box.

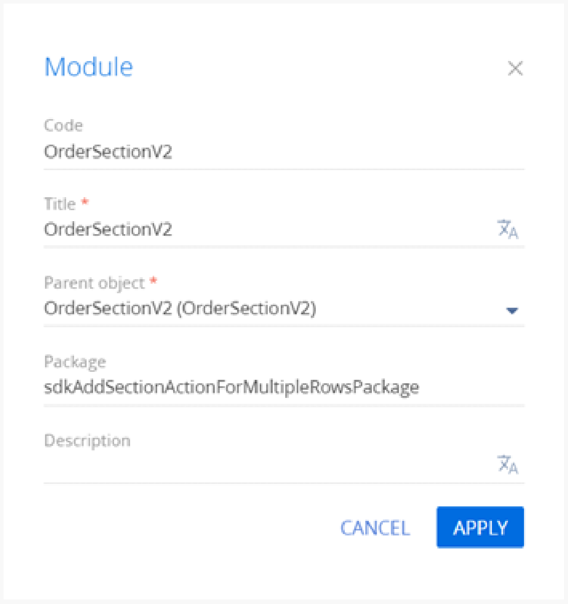
Create a schema of the replacing section view model

1. Go to the [*Configuration*] section and select a custom package to add the schema.
2. Click [*Add*] → [*Replacing view model*] on the section list toolbar.



3. Fill out the **schema properties**.

- Set [*Code*] to "`OrderSectionV2`".
- Set [*Title*] to "`Order section`".
- Set [*Parent object*] to "`OrderSectionV2`".



Code
OrderSectionV2

Title *
OrderSectionV2

Parent object *
OrderSectionV2 (OrderSectionV2)

Package
sdkAddSectionActionForMultipleRowsPackage

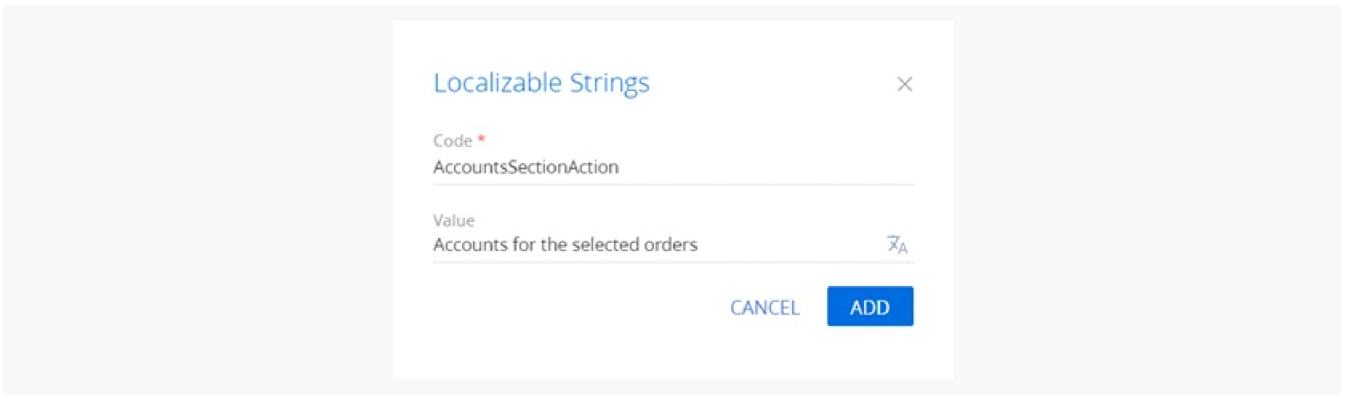
Description

4. Add a **localizable string** that contains the menu item caption.

   a. Click the + button in the context menu of the [*Localizable strings*] node.

   b. Fill out the **localizable string properties**.

   - Set [*Code*] to "`AccountsSectionAction`".
   - Set [*Value*] to "`Accounts for the selected orders`".



Code *
AccountsSectionAction

Value
Accounts for the selected orders

e. Click [*Add*] to add a localizable string.

5. Implement the **menu item behavior**. To do this, implement the following **methods** in the `methods` property:

- `isCustomActionEnabled()`. Determines whether the menu item is available.
- `showOrderInfo()`. The action handler method that displays the index of accounts for the selected orders in the message box.
- `getSectionActions()`. The overloaded parent schema method that retrieves the section's action collection.

The primary column values of the selected records are stored in the `SelectedRows` property of the section view model. You can use these values to get the values uploaded to the field list of the selected objects. For example, from a regular list's data collection stored in the `GridData` view model property of the list.

View the source code of the replacing view model schema of the section below.

`OrderSectionV2`

```
define("OrderSectionV2", ["OrderConfigurationConstants"], function(OrderConfigurationConstant
    return {
        /* The name of the section object schema. */
        entitySchemaName: "Order",
        /* The methods of the section view model. */
        methods: {
            /* Determine whether the menu item is available. */
            isCustomActionEnabled: function() {
                /* Attempt to retrieve the array of the selected record IDs. */
                var selectedRows = this.get("SelectedRows");
                /* Return true if the array contains items (at least one list record is selec
                return selectedRows ? (selectedRows.length > 0) : false;
            },
            /* The action handler method. Displays the index of accounts in the message box.
            showOrdersInfo: function() {
                /* Retrieve the array of the selected record IDs. */
                var selectedRows = this.get("SelectedRows");
                /* Retrieve the data collection of the list records. */
                var gridData = this.get("GridData");
                /* The variable that stores the object model of the selected order. */
                var selectedOrder = null;
                /* The variable that stores the account name for the selected order. */
                var selectedOrderAccount = "";
                /* The variable that generates the message box text. */
                var infoText = "";
                /* Process the array of the selected record IDs. */
                selectedRows.forEach(function(selectedRowId) {
                    /* Retrieve the object model of the selected order. */
                    selectedOrder = gridData.get(selectedRowId);
                    /* Retrieve the account name for the selected order. The column must be a
                    selectedOrderAccount = selectedOrder.get("Account").displayValue;
                    /* Add the account name to the message box text. */
                    infoText += "\n" + selectedOrderAccount;
                });
                /* Display the message box. */
                this.showInformationDialog(infoText);
            },
            /* Overload the base virtual method that returns the section action collection. *
            getSectionActions: function() {
                /* Call the parent method implementation that retrieves the collection of the
                var actionMenuItems = this.callParent(arguments);
                /* Add a separator line. */
                actionMenuItems.addItem(this.getButtonMenuItem({
                    Type: "Terrasoft.MenuSeparator",
                    Caption: ""
                }));
                /* Add a menu item to the section action index. */
                actionMenuItems.addItem(this.getButtonMenuItem({
                    /* Bind the menu item caption to the localizable schema string. */
                    "Caption": {bindTo: "Resources.Strings.AccountsSectionAction"},
                    /* Bind the action handler method. */
                    "Click": {bindTo: "showOrdersInfo"},
                    /* Bind the property of the menu item availability to the value returned
                    "Enabled": {bindTo: "isCustomActionEnabled"},
                    /* Support the multiple selection mode. */
                    "IsEnabledForSelectedAll": true
                }));
                /* Return the expanded section action collection. */
                return actionMenuItems;
            }
        }
    };
});
```

6. Click [*Save*] on the Designer's toolbar.

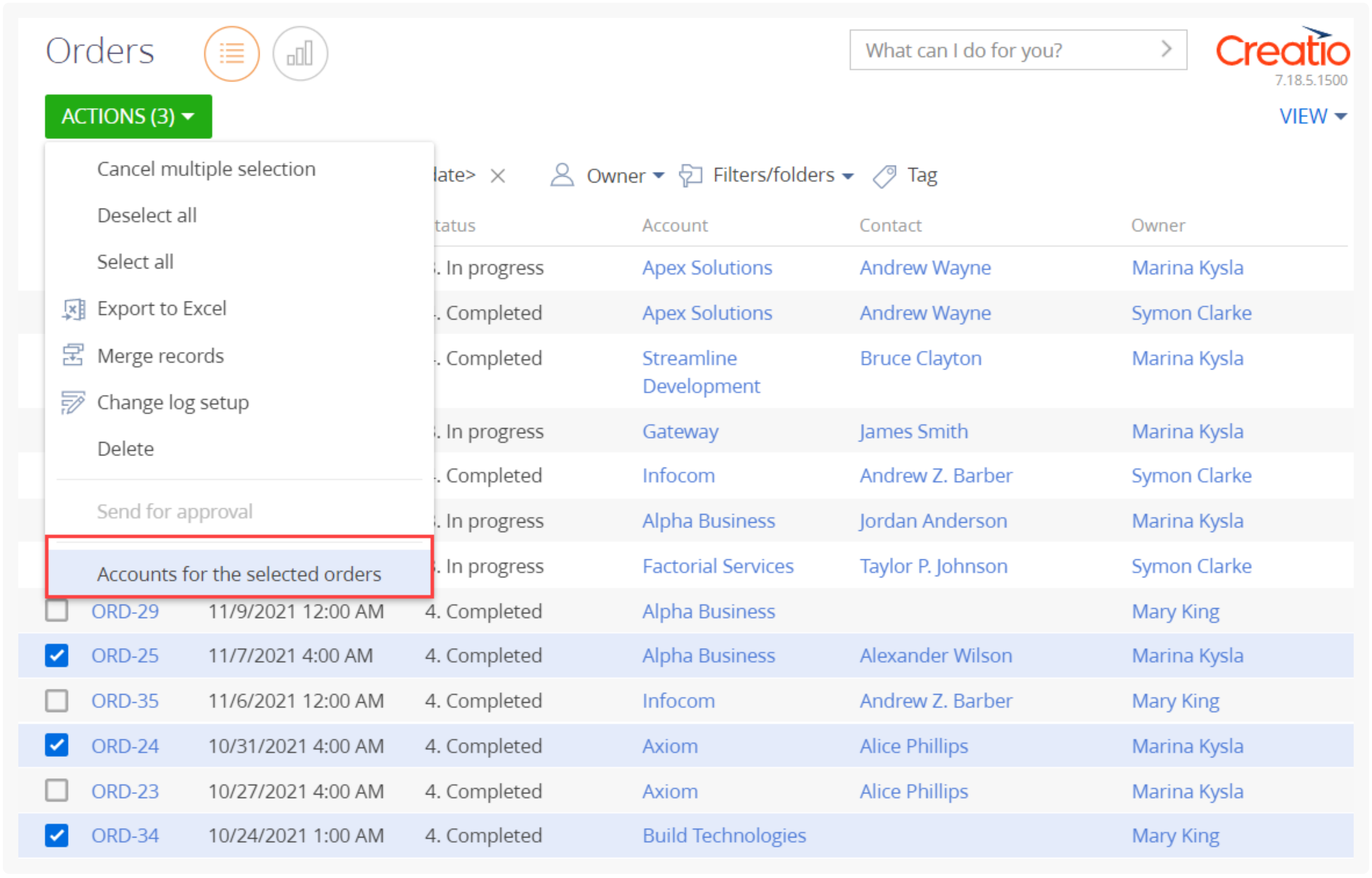
The outcome of the example

To **view the outcome of the example**:

1. Clear the browser cache.
2. Refresh the [*Orders*] section page.

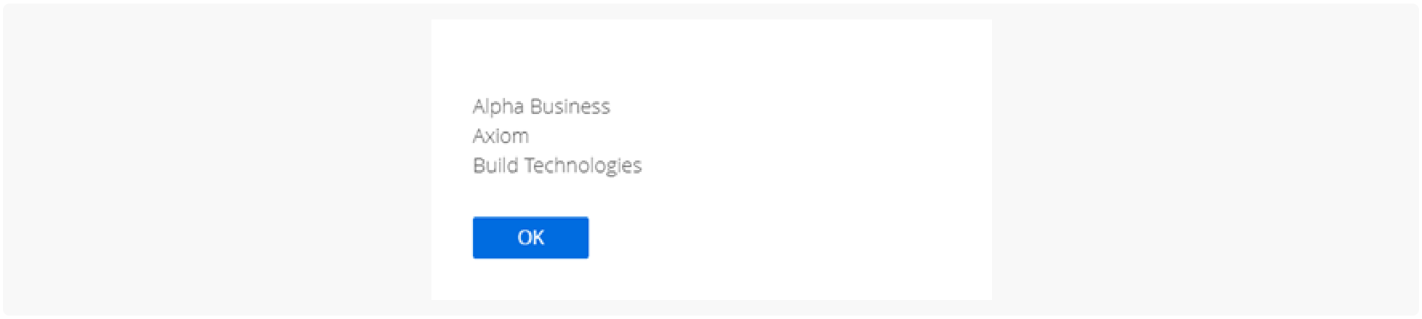
The section single record mode is used by default. To select multiple list records, click [*Select multiple records*] in the menu of the [*Actions*] button. This changes the visual view of the list. You will see record selection elements appear.

As a result, Creatio will add the [*Accounts for the selected orders*] action to the order page.



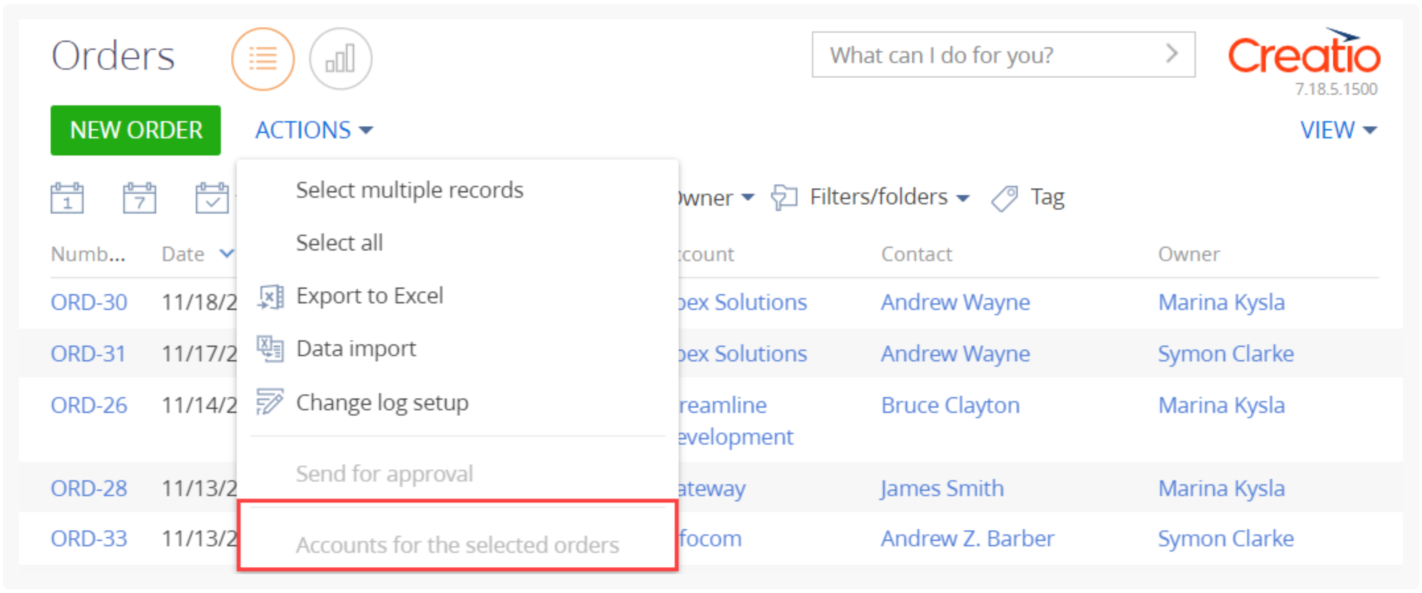
			tatus	Account	Contact	Owner
			. In progress	Apex Solutions	Andrew Wayne	Marina Kysla
			. Completed	Apex Solutions	Andrew Wayne	Symon Clarke
			. Completed	Streamline Development	Bruce Clayton	Marina Kysla
			. In progress	Gateway	James Smith	Marina Kysla
			. Completed	Infocom	Andrew Z. Barber	Symon Clarke
			. In progress	Alpha Business	Jordan Anderson	Marina Kysla
			. In progress	Factorial Services	Taylor P. Johnson	Symon Clarke
☐	ORD-29	11/9/2021 12:00 AM	4. Completed	Alpha Business		Mary King
✓	ORD-25	11/7/2021 4:00 AM	4. Completed	Alpha Business	Alexander Wilson	Marina Kysla
☐	ORD-35	11/6/2021 12:00 AM	4. Completed	Infocom	Andrew Z. Barber	Mary King
✓	ORD-24	10/31/2021 4:00 AM	4. Completed	Axiom	Alice Phillips	Marina Kysla
☐	ORD-23	10/27/2021 4:00 AM	4. Completed	Axiom	Alice Phillips	Marina Kysla
✓	ORD-34	10/24/2021 1:00 AM	4. Completed	Build Technologies		Mary King

Run the [*Accounts for the selected orders*] action to bring up the message box that displays the account list for the selected orders.


Alpha Business
Axiom
Build Technologies

To cancel the multiple selection mode, click [*Cancel multiple selection*] in the menu of the [*Actions*] button.

If you do not select any records in the [*Orders*] section list, the [*Accounts for the selected orders*] action will be inactive.



Numb...	Date	Account	Contact	Owner
ORD-30	11/18/2	ɔex Solutions	Andrew Wayne	Marina Kysla
ORD-31	11/17/2	ɔex Solutions	Andrew Wayne	Symon Clarke
ORD-26	11/14/2	reamline evelopment	Bruce Clayton	Marina Kysla
ORD-28	11/13/2	ateway	James Smith	Marina Kysla
ORD-33	11/13/2	focom	Andrew Z. Barber	Symon Clarke

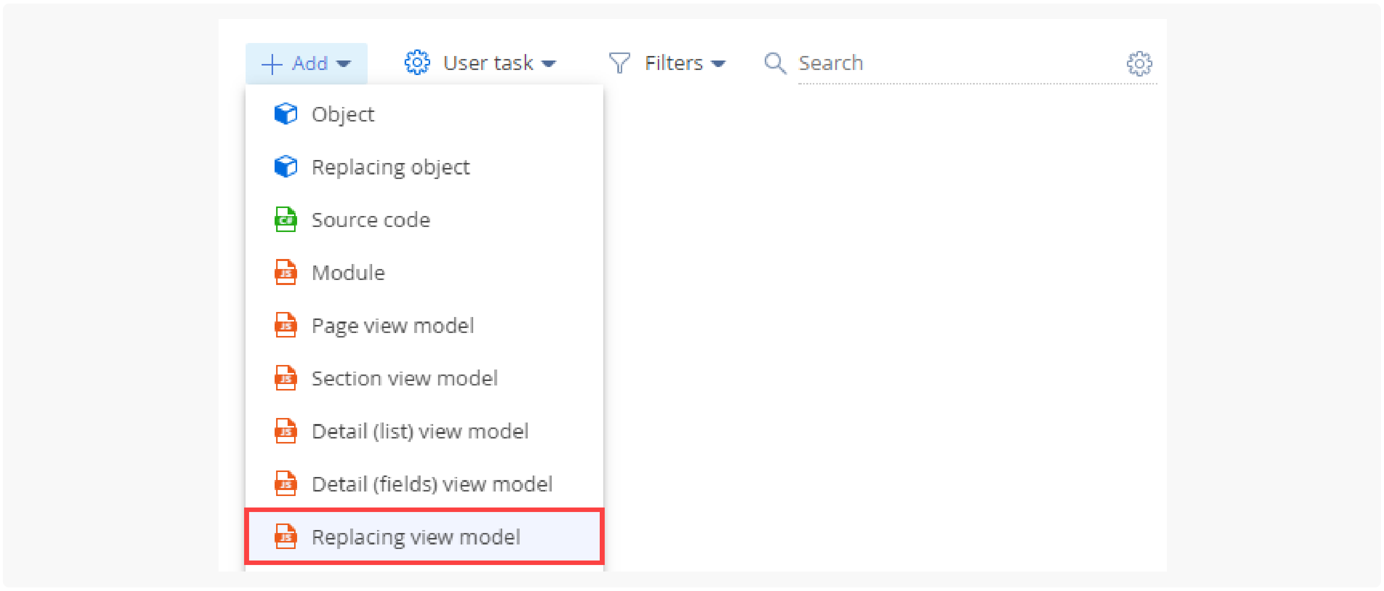
Set up action handling for multiple section records

Medium

Example. Set up action handling in the [*Activities*] section list. The action sets the [*Completed*] status for selected activities.

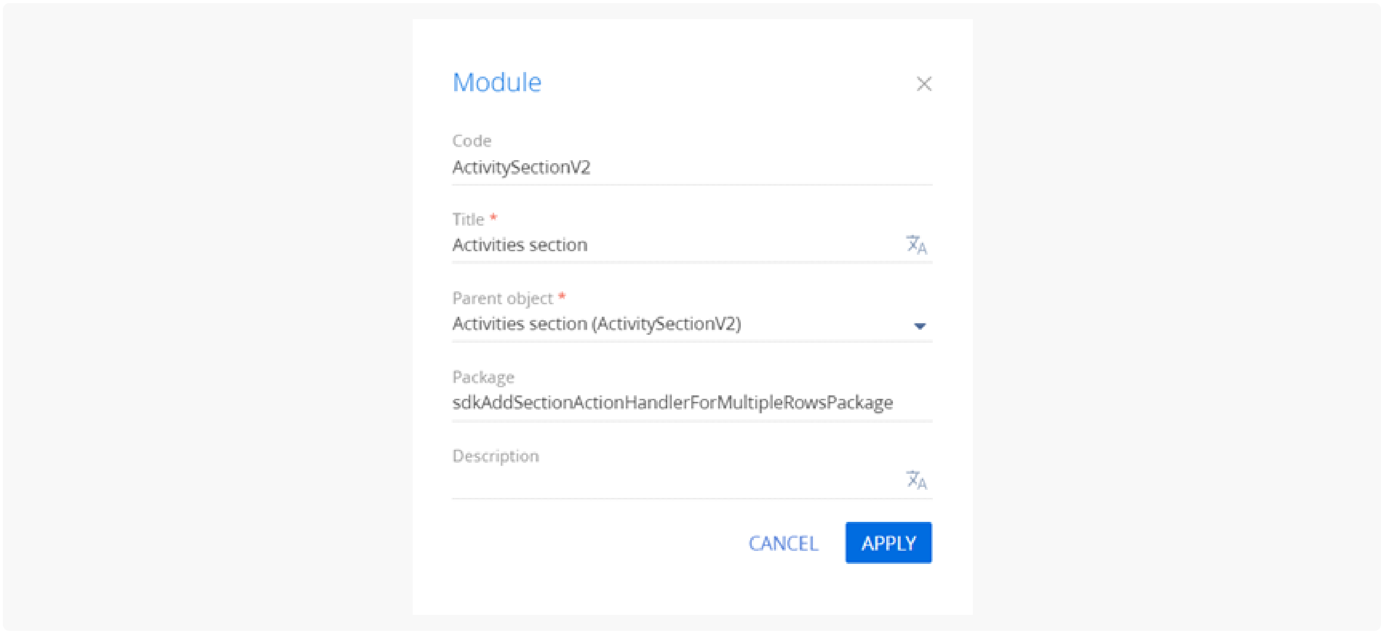
Create a schema of the replacing section view model

1. Go to the [*Configuration*] section and select a custom package to add the schema.
2. Click [*Add*] → [*Replacing view model*] on the section list toolbar.



3. Fill out the **schema properties**.

- Set [*Code*] to "`ActivitySectionV2`".
- Set [*Title*] to "`Activities section`".
- Set [*Parent object*] to "`ActivitySectionV2`".



Code
ActivitySectionV2

Title *
Activities section

Parent object *
Activities section (ActivitySectionV2)

Package
sdkAddSectionActionHandlerForMultipleRowsPackage

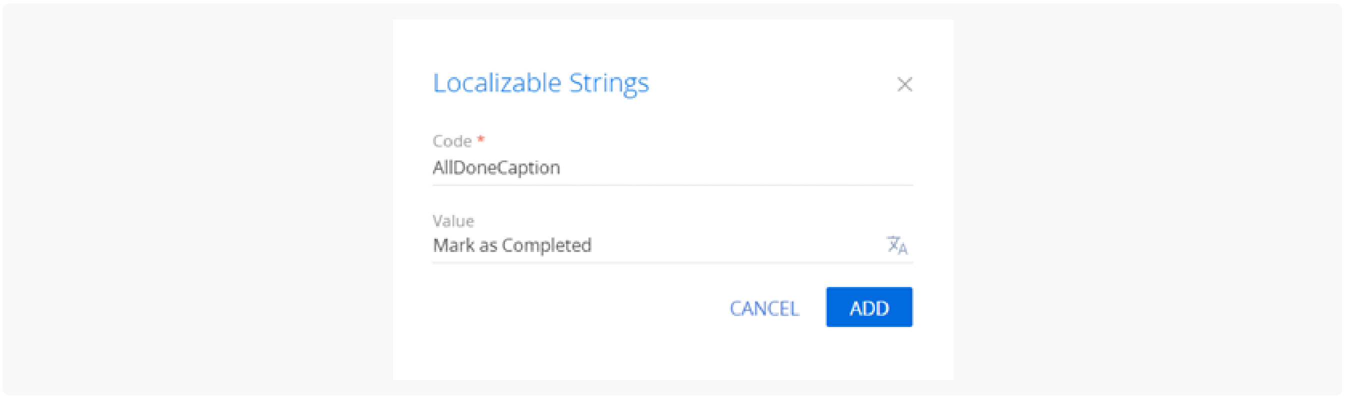
Description

4. Add a **localizable string** that contains the menu item caption.

   a. Click the + button in the context menu of the [*Localizable strings*] node.

   b. Fill out the **localizable string properties**.

   - Set [*Code*] to "`AllDoneCaption`".
   - Set [*Value*] to "`Mark as Completed`".



Code *
AllDoneCaption

Value
Mark as Completed

e. Click [*Add*] to add a localizable string.

5. Implement the **menu item behavior**. To do this, implement the following **methods** in the `methods` property:

- `isCustomActionEnabled()`. Determines whether the menu item is available.
- `setAllDone()`. The action handler method that sets the [*Completed*] status for selected activities.
- `getSectionActions()`. The overloaded parent schema method that retrieves the section action collection.

View the source code of the replacing view model schema of the section below.

ActivitySectionV2

```
define("ActivitySectionV2", ["ConfigurationConstants"], function(ConfigurationConstants) {
    return {
        /* The name of the section object schema. */
        entitySchemaName: "Activity",
        /* The methods of the section view model. */
        methods: {
            /* Determine whether the menu item is available. */
            isCustomActionEnabled: function() {
                /* Attempt to retrieve the array of the selected record IDs. */
                var selectedRows = this.get("SelectedRows");
                /* Return true if the array contains items (at least one list record is selec
                return selectedRows ? (selectedRows.length > 0) : false;
            },
            /* The action handler method. Sets the [Completed] status for selected records. *
            setAllDone: function() {
                /* Retrieve the array of the selected record IDs. */
                var selectedRows = this.get("SelectedRows");
                /* Run handling if at least one record was selected. */
                if (selectedRows.length > 0) {
                    /* Create an instance of the batch query class. */
                    var batchQuery = this.Ext.create("Terrasoft.BatchQuery");
                    /* Update each of the selected records. */
                    selectedRows.forEach(function(selectedRowId) {
                        /* Create an UpdateQuery class instance with the Activity root schema
                        var update = this.Ext.create("Terrasoft.UpdateQuery", {
                            rootSchemaName: "Activity"
                        });
                        /* Apply the filter to determine the record to update. */
                        update.enablePrimaryColumnFilter(selectedRowId);
                        /* Set the [Status] column to "Completed" using the ConfigurationCons
                        update.setParameterValue("Status", ConfigurationConstants.Activity.St
                        /* Add a record update query to the batch query. */
                        batchQuery.add(update);
                    }, this);
                    /* Execute the batch server query. */
                    batchQuery.execute(function() {
                        /* Update the list. */
                        this.reloadGridData();
                    }, this);
                }
            },
            /* Overload the base virtual method that returns the section action collection. *
            getSectionActions: function() {
                /* Call the parent method implementation that retrieves the collection of the
                var actionMenuItems = this.callParent(arguments);
                /* Add a separator line. */
                actionMenuItems.addItem(this.getButtonMenuItem({
                    Type: "Terrasoft.MenuSeparator",
                    Caption: ""
                }));
                /* Add a menu item to the section action index. */
                actionMenuItems.addItem(this.getButtonMenuItem({
                    /* Bind the menu item caption to the localizable schema string. */
                    "Caption": { bindTo: "Resources.Strings.AllDoneCaption" },
                    /* Bind the action handler method. */
                    "Click": { bindTo: "setAllDone" },
                    /* Bind the property of the menu item availability to the value returned
                    "Enabled": { bindTo: "isCustomActionEnabled" },
                    /* Support the multiple selection mode. */
                    "IsEnabledForSelectedAll": true
                }));
                /* Return the expanded section action collection. */
                return actionMenuItems;
            }
        }
    };
});
```

6. Click [*Save*] on the Designer's toolbar.

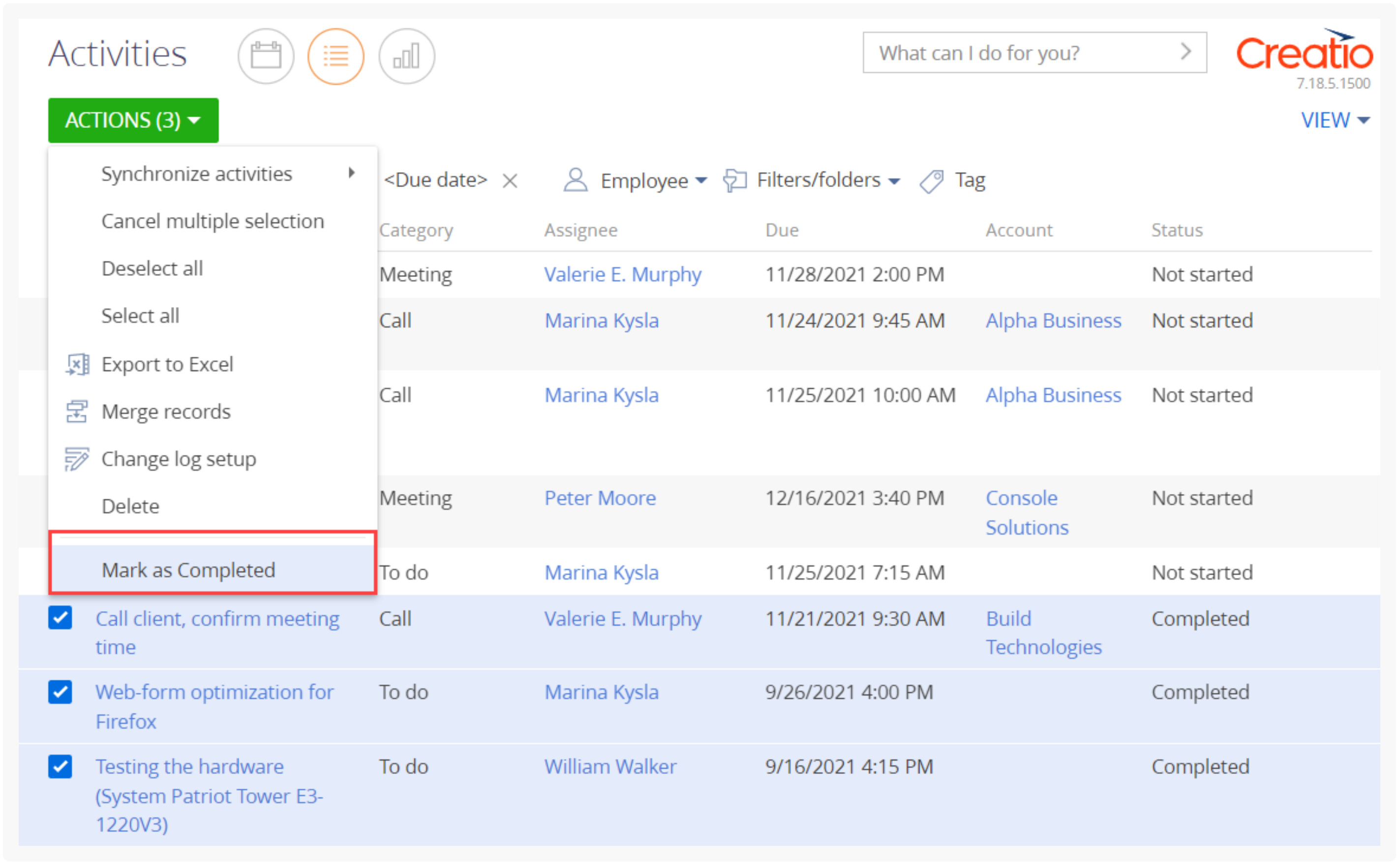
The outcome of the example

To **view the outcome of the example**:

1. Clear the browser cache.
2. Refresh the [*Activities*] section page.

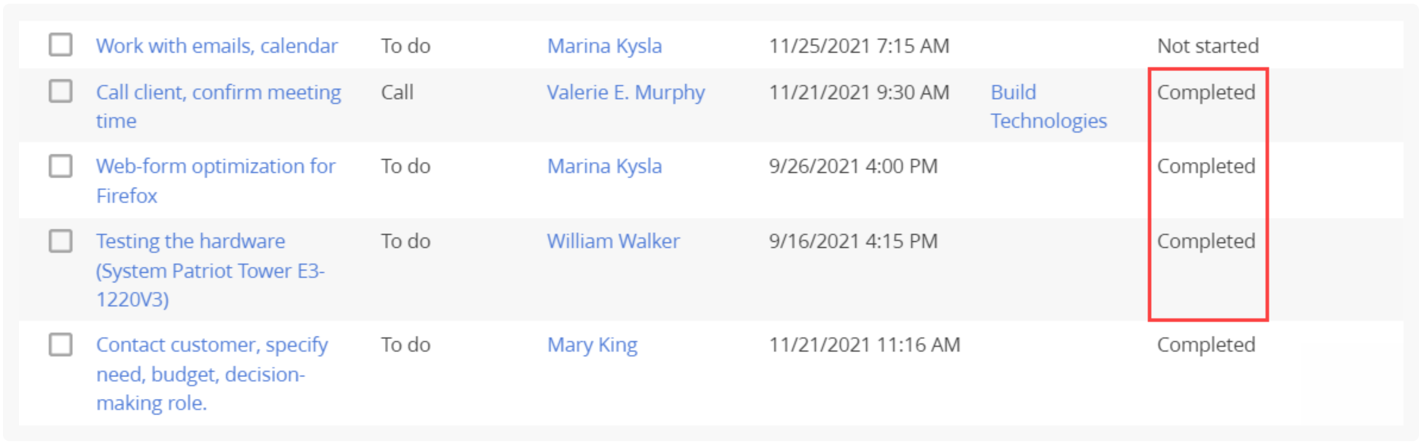
The section single record mode is used by default. To select multiple list records, click [*Select multiple records*] in the menu of the [*Actions*] button. This changes the visual view of the list. You will see record selection elements appear.

As a result, Creatio will add the [*Mark as Completed*] action to the activity page.



	Category	Assignee	Due	Account	Status
	Meeting	Valerie E. Murphy	11/28/2021 2:00 PM		Not started
	Call	Marina Kysla	11/24/2021 9:45 AM	Alpha Business	Not started
	Call	Marina Kysla	11/25/2021 10:00 AM	Alpha Business	Not started
	Meeting	Peter Moore	12/16/2021 3:40 PM	Console Solutions	Not started
	To do	Marina Kysla	11/25/2021 7:15 AM		Not started
✓ Call client, confirm meeting time	Call	Valerie E. Murphy	11/21/2021 9:30 AM	Build Technologies	Completed
✓ Web-form optimization for Firefox	To do	Marina Kysla	9/26/2021 4:00 PM		Completed
✓ Testing the hardware (System Patriot Tower E3-1220V3)	To do	William Walker	9/16/2021 4:15 PM		Completed

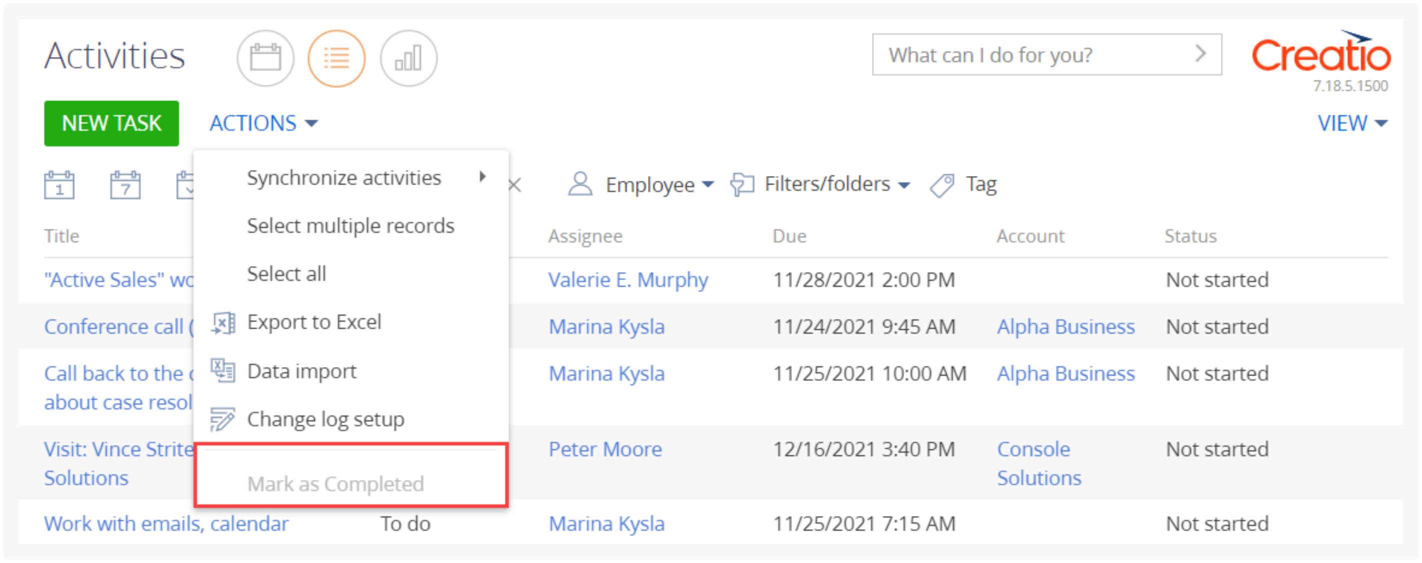
Run the [*Mark as Completed*] action to set the [*Status*] column of the selected activities to [*Completed*].



☐	Work with emails, calendar	To do	Marina Kysla	11/25/2021 7:15 AM		Not started
☐	Call client, confirm meeting time	Call	Valerie E. Murphy	11/21/2021 9:30 AM	Build Technologies	Completed
☐	Web-form optimization for Firefox	To do	Marina Kysla	9/26/2021 4:00 PM		Completed
☐	Testing the hardware (System Patriot Tower E3-1220V3)	To do	William Walker	9/16/2021 4:15 PM		Completed
☐	Contact customer, specify need, budget, decision-making role.	To do	Mary King	11/21/2021 11:16 AM		Completed

To cancel the multiple selection mode, click [*Cancel multiple selection*] in the menu of the [*Actions*] button.

If you do not select any records in the [*Activities*] section list, the [*Mark as Completed*] action will be inactive.



Title		Assignee	Due	Account	Status
"Active Sales" wo		Valerie E. Murphy	11/28/2021 2:00 PM		Not started
Conference call (		Marina Kysla	11/24/2021 9:45 AM	Alpha Business	Not started
Call back to the c about case resol		Marina Kysla	11/25/2021 10:00 AM	Alpha Business	Not started
Visit: Vince Strite Solutions		Peter Moore	12/16/2021 3:40 PM	Console Solutions	Not started
Work with emails, calendar	To do	Marina Kysla	11/25/2021 7:15 AM		Not started

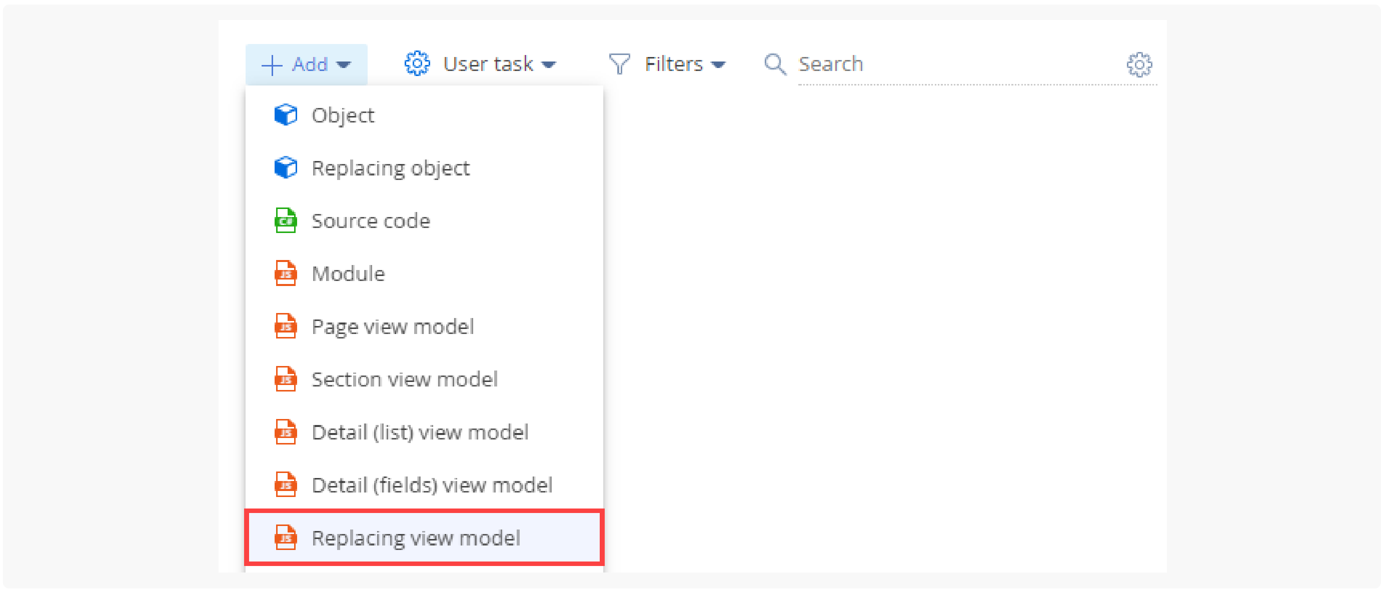
Set up action handling using a lookup for multiple section records

Medium

Example. Set up action handling in the [*Activities*] section list. The action must open the [*Contacts*] lookup box. Set the selected contact as the owner of the selected activities.

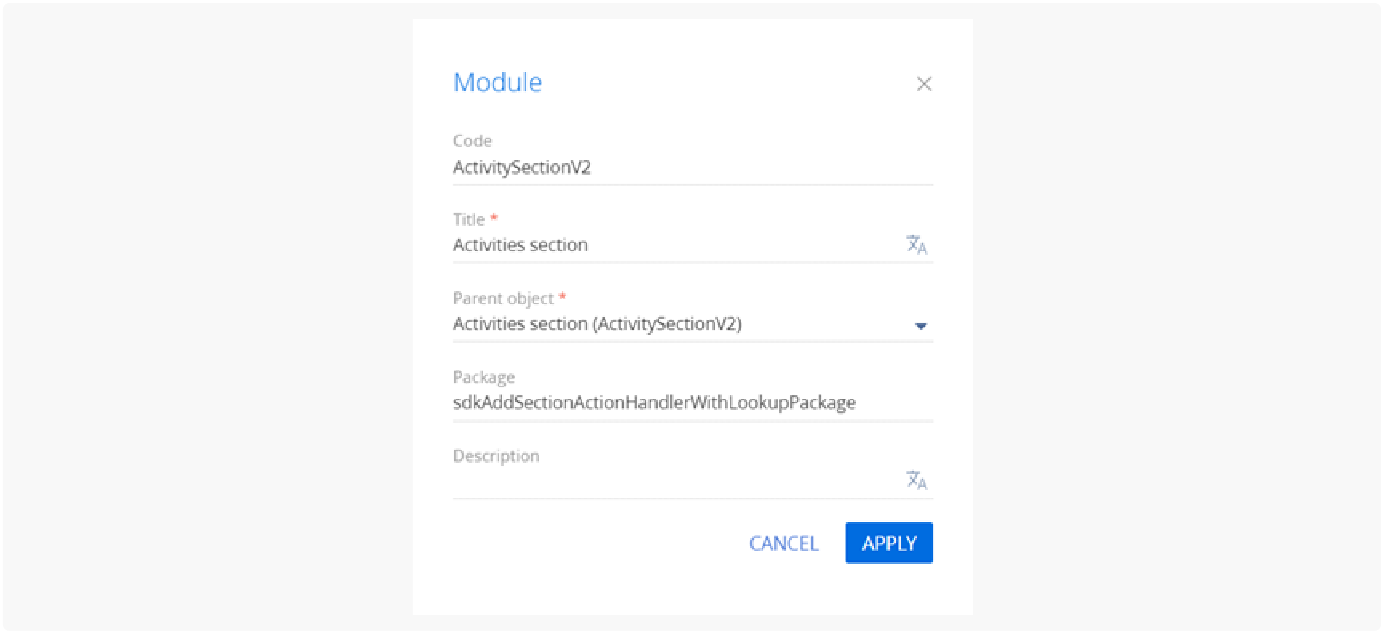
Create a schema of the replacing section view model

1. Go to the [*Configuration*] section and select a custom package to add the schema.
2. Click [*Add*] → [*Replacing view model*] on the section list toolbar.



3. Fill out the **schema properties**.

- Set [*Code*] to "`ActivitySectionV2`".
- Set [*Title*] to "`Activities section`".
- Set [*Parent object*] to "`ActivitySectionV2`".


Module

Code
ActivitySectionV2

Title *
Activities section

Parent object *
Activities section (ActivitySectionV2)

Package
sdkAddSectionActionHandlerWithLookupPackage

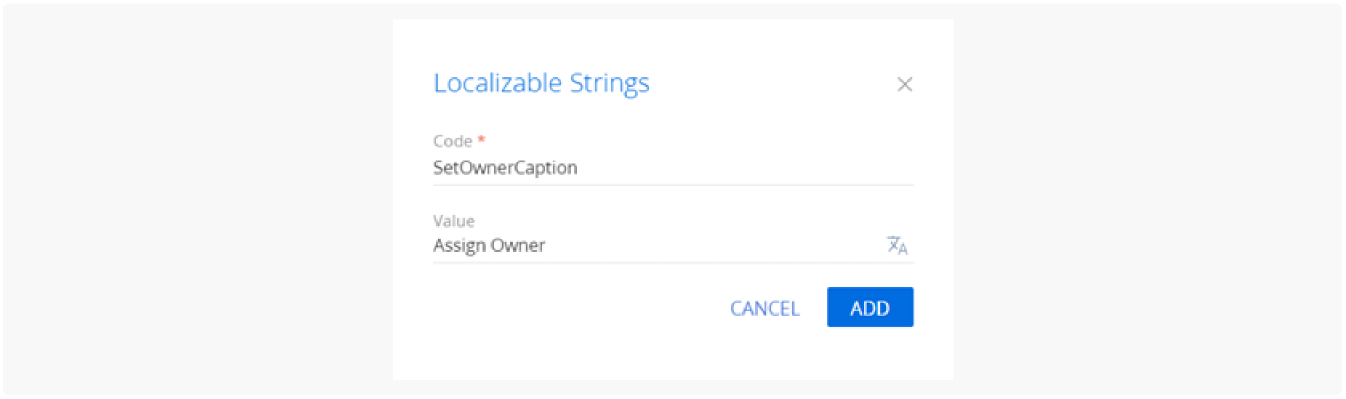
Description

4. Add a **localizable string** that contains the menu item caption.

   a. Click the + button in the context menu of the [*Localizable strings*] node.

   b. Fill out the **localizable string properties**.

   - Set [*Code*] to "`SetOwnerCaption`".
   - Set [*Value*] to "`Assign Owner`".



Code *
SetOwnerCaption

Value
Assign Owner

e. Click [*Add*] to add a localizable string.

5. Implement the **menu item behavior**. To do this, implement the following **methods** in the `methods` property:

- `isCustomActionEnabled()`. Determines whether the menu item is available.
- `setOwner()`. The action handler method that opens the [*Contacts*] lookup.
- `lookupCallback()`. The callback method that sets the contact selected from the lookup as the owner of the selected activities.
- `getSectionActions()`. The overloaded parent schema method that retrieves the section action collection.

View the source code of the replacing view model schema of the section below.

ActivitySectionV2

```
define("ActivitySectionV2", ["ConfigurationConstants"], function(ConfigurationConstants) {
    return {
        /* The name of the section object schema. */
        entitySchemaName: "Activity",
        /* The methods of the section view model. */
        methods: {
            /* Determine whether the menu item is available. */
            isCustomActionEnabled: function() {
                /* Attempt to retrieve the array of the selected record IDs. */
                var selectedRows = this.get("SelectedRows");
                /* Return true if the array contains items (at least one list record is selec
                return selectedRows ? (selectedRows.length > 0) : false;
            },
            /* The action handler method. Opens the [Contacts] lookup. */
            setOwner: function() {
                /* Define the lookup configuration. */
                var config = {
                    /* The [Contact] schema. */
                    entitySchemaName: "Contact",
                    /* Disable multiple selection. */
                    multiSelect: false,
                    /* Display the [Name] column. */
                    columns: ["Name"]
                };
                /* Open the lookup with a particular configuration and the callback function
                this.openLookup(config, this.lookupCallback, this);
            },
            /* Set the contact selected from the lookup as the owner of the selected activiti
            lookupCallback: function(args) {
                /* The ID of the selected lookup record. */
                var activeRowId;
                /* Retrieve the selected lookup records. */
                var lookupSelectedRows = args.selectedRows.getItems();
                if (lookupSelectedRows && lookupSelectedRows.length > 0) {
                    /* Retrieve the ID of the first selected lookup record. */
                    activeRowId = lookupSelectedRows[0].Id;
                }
                /* Retrieve the array of the selected record IDs. */
                var selectedRows = this.get("SelectedRows");
                /* Run handling if at least one record was selected and the owner was selecte
                if ((selectedRows.length > 0) && activeRowId) {
                    /* Create an instance of the batch query class. */
                    var batchQuery = this.Ext.create("Terrasoft.BatchQuery");
                    /* Update each of the selected records. */
                    selectedRows.forEach(function(selectedRowId) {
                        /* Create an UpdateQuery class instance with the Activity root schema
                        var update = this.Ext.create("Terrasoft.UpdateQuery", {
                            rootSchemaName: "Activity"
                        });
                        /* Apply the filter to determine the record to update. */
                        update.enablePrimaryColumnFilter(selectedRowId);
                        /* Set the [Owner] column to the ID of the contact selected from the
                        update.setParameterValue("Owner", activeRowId, this.Terrasoft.DataVal
                        /* Add a record update query to the batch query. */
                        batchQuery.add(update);
                    }, this);
                    /* Execute the batch server query. */
                    batchQuery.execute(function() {
                        /* Update the list. */
                        this.reloadGridData();
                    }, this);
                }
            },
            /* Overload the base virtual method that returns the section action collection. *
            getSectionActions: function() {
                /* Call the parent method implementation that retrieves the collection of the
                var actionMenuItems = this.callParent(arguments);
                /* Add a separator line. */
                actionMenuItems.addItem(this.getButtonMenuItem({
                    Type: "Terrasoft.MenuSeparator",
                    Caption: ""
                }));
                /* Add a menu item to the section action index. */
                actionMenuItems.addItem(this.getButtonMenuItem({
                    /* Bind the menu item caption to the localizable schema string. */
                    "Caption": { bindTo: "Resources.Strings.SetOwnerCaption" },
                    /* Bind the action handler method. */
                    "Click": { bindTo: "setOwner" },
                    /* Bind the property of the menu item availability to the value returned
                    "Enabled": { bindTo: "isCustomActionEnabled" },
                    /* Support the multiple selection mode. */
                    "IsEnabledForSelectedAll": true
                }));
                /* Return the expanded section action collection. */
                return actionMenuItems;
            }
        }
    };
});
```

6. Click [*Save*] on the Designer&'s toolbar.

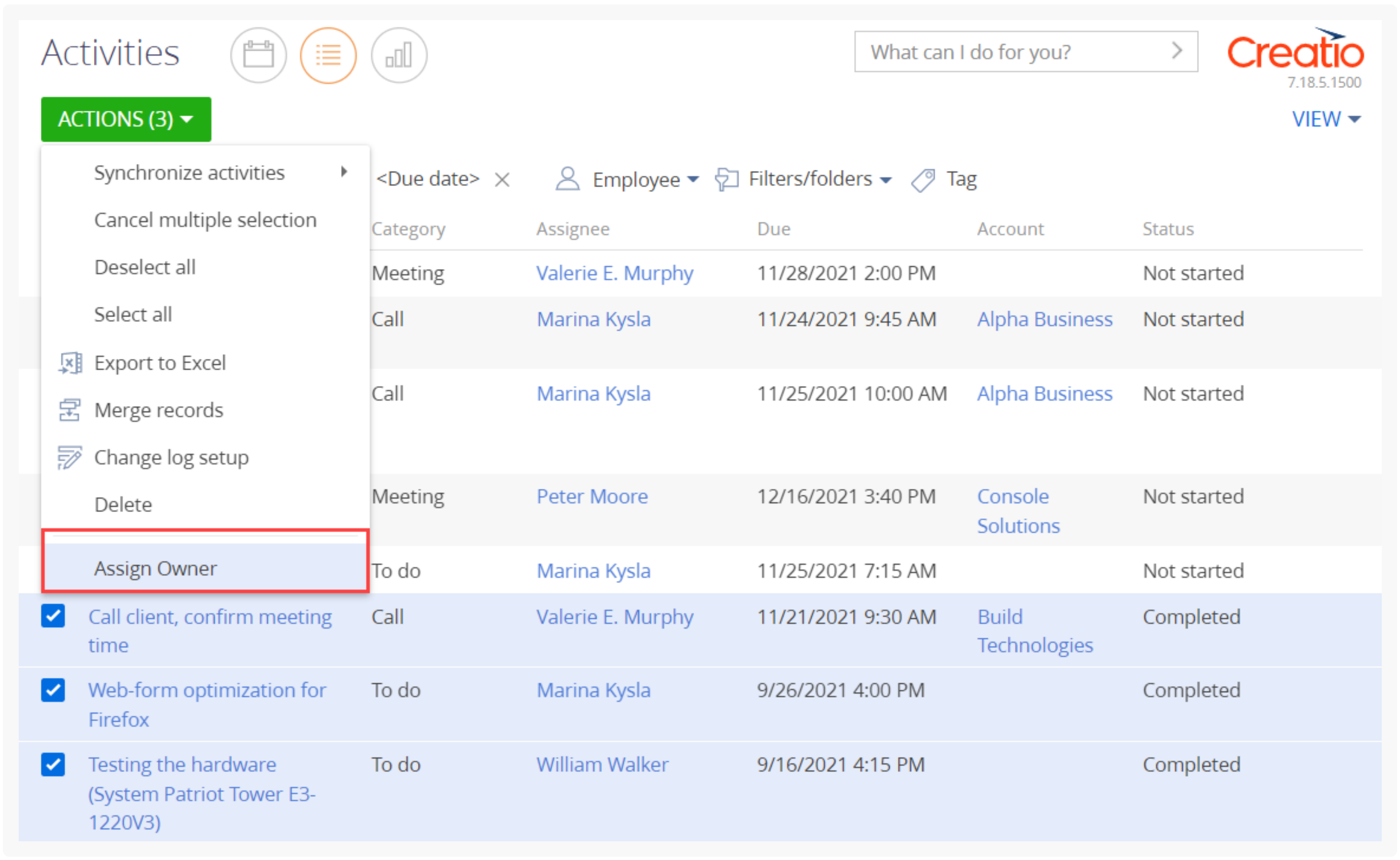
The outcome of the example

To **view the outcome of the example**:

1. Clear the browser cache.
2. Refresh the [*Activities*] section page.

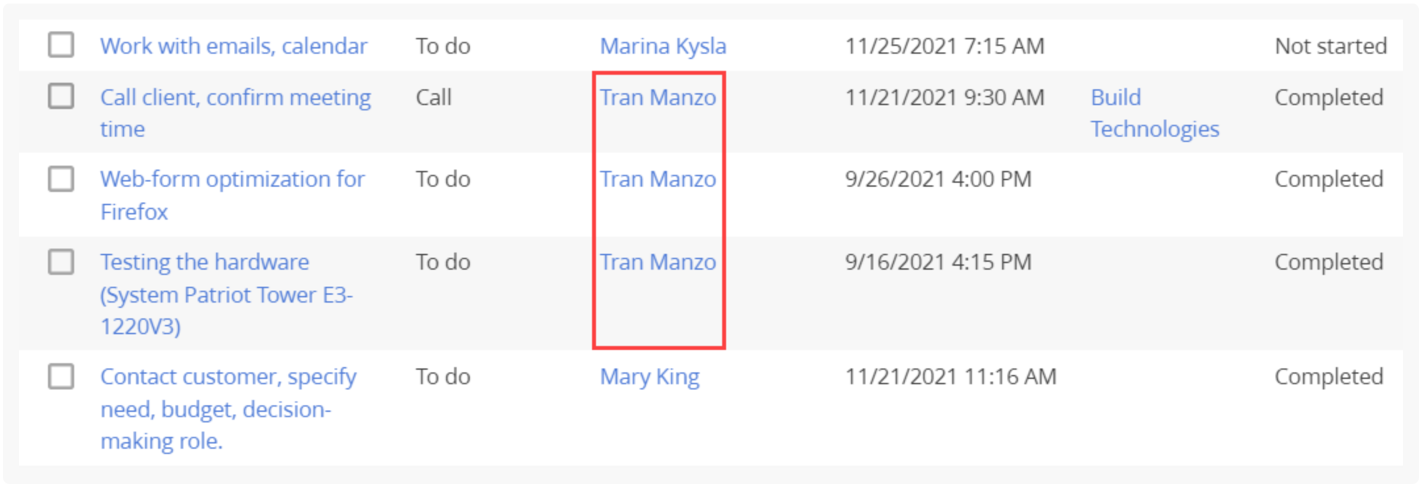
The section single record mode is used by default. To select multiple list records, click [*Select multiple records*] in the menu of the [*Actions*] button. This changes the visual view of the list. You will see record selection elements appear.

As a result, Creatio will add the [*Assign Owner*] action to the order page.



	Category	Assignee	Due	Account	Status
	Meeting	Valerie E. Murphy	11/28/2021 2:00 PM		Not started
	Call	Marina Kysla	11/24/2021 9:45 AM	Alpha Business	Not started
	Call	Marina Kysla	11/25/2021 10:00 AM	Alpha Business	Not started
	Meeting	Peter Moore	12/16/2021 3:40 PM	Console Solutions	Not started
	To do	Marina Kysla	11/25/2021 7:15 AM		Not started
☑ Call client, confirm meeting time	Call	Valerie E. Murphy	11/21/2021 9:30 AM	Build Technologies	Completed
☑ Web-form optimization for Firefox	To do	Marina Kysla	9/26/2021 4:00 PM		Completed
☑ Testing the hardware (System Patriot Tower E3-1220V3)	To do	William Walker	9/16/2021 4:15 PM		Completed

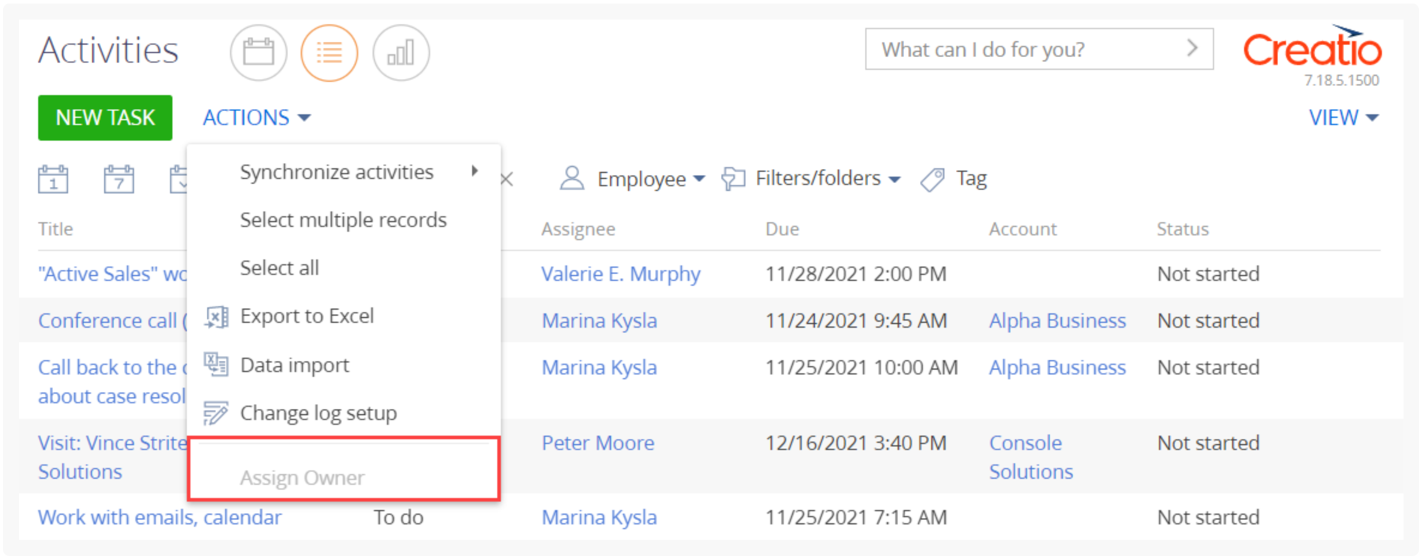
Run the [*Assign Owner*] action to open the [*Contacts*] lookup box. Creatio sets the selected contact as the owner of the selected activities in the [*Assignee*] column.



☐ Work with emails, calendar	To do	Marina Kysla	11/25/2021 7:15 AM		Not started
☐ Call client, confirm meeting time	Call	Tran Manzo	11/21/2021 9:30 AM	Build Technologies	Completed
☐ Web-form optimization for Firefox	To do	Tran Manzo	9/26/2021 4:00 PM		Completed
☐ Testing the hardware (System Patriot Tower E3-1220V3)	To do	Tran Manzo	9/16/2021 4:15 PM		Completed
☐ Contact customer, specify need, budget, decision-making role.	To do	Mary King	11/21/2021 11:16 AM		Completed

To cancel the multiple selection mode, click [*Cancel multiple selection*] in the menu of the [*Actions*] button.

If you do not select any records in the [*Activities*] section list, the [*Assign Owner*] action will be inactive.



Title		Assignee	Due	Account	Status
"Active Sales" wo		Valerie E. Murphy	11/28/2021 2:00 PM		Not started
Conference call (		Marina Kysla	11/24/2021 9:45 AM	Alpha Business	Not started
Call back to the c about case resol		Marina Kysla	11/25/2021 10:00 AM	Alpha Business	Not started
Visit: Vince Strite Solutions		Peter Moore	12/16/2021 3:40 PM	Console Solutions	Not started
Work with emails, calendar	To do	Marina Kysla	11/25/2021 7:15 AM		Not started

Set up the display conditions of the section list

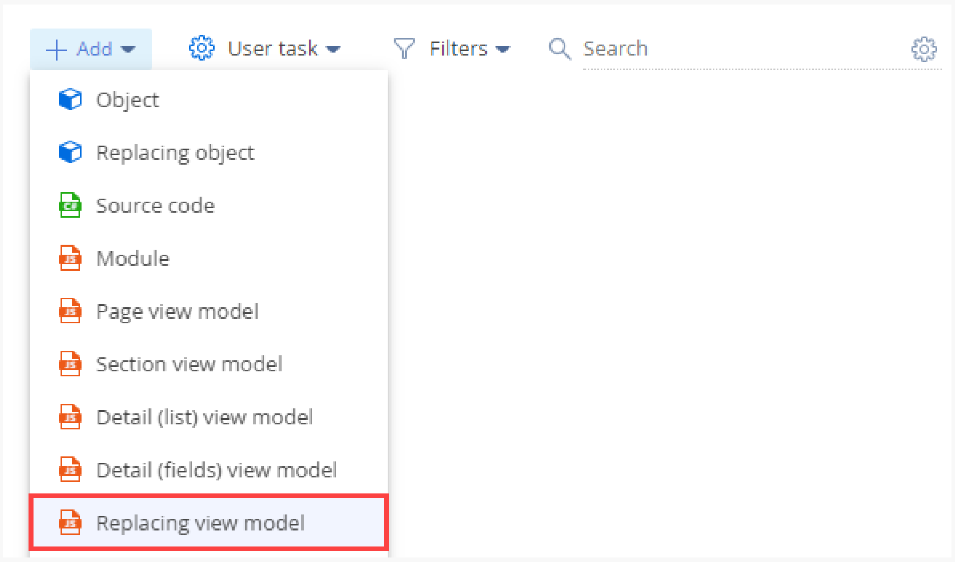
Medium

The example is relevant to Sales Creatio products.

Example. In the [*Orders*] section list, highlight orders at the [*In progress*] stage green.

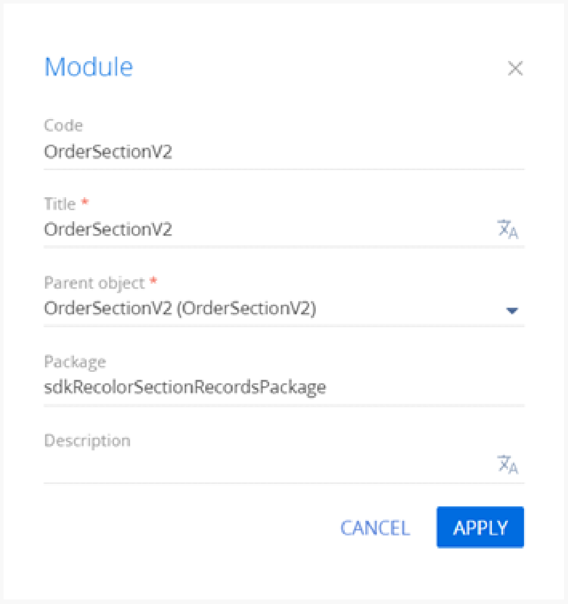
Create a schema of the replacing section view model

1. Go to the [*Configuration*] section and select a custom package to add the schema.
2. Click [*Add*] → [*Replacing view model*] on the section list toolbar.



3. Fill out the **schema properties**.

- Set [*Code*] to "`OrderSectionV2`".
- Set [*Title*] to "`Order section`".
- Set [*Parent object*] to "`OrderSectionV2`".



Code
OrderSectionV2

Title *
OrderSectionV2

Parent object *
OrderSectionV2 (OrderSectionV2)

Package
sdkRecolorSectionRecordsPackage

Description

4. Set up the **appearance conditions of the section list**. To do this, implement the `prepareResponseCollectionItem()` method in the `methods` property. The method overloads the base method, modifies the data row before uploading the row to the list, as well as injects custom styles into specific list rows.

   View the source code of the replacing view model schema of the section below.

`OrderSectionV2`

```
define("OrderSectionV2", ["OrderConfigurationConstants"], function(OrderConfigurationConstant
    return {
        /* The name of the section object schema. */
        entitySchemaName: "Order",
        /* The methods of the section view model. */
        methods: {
            /* Overload the base method that modifies the data row before uploading the row t
            prepareResponseCollectionItem: function(item) {
                /* Call the base method. */
                this.callParent(arguments);
                item.customStyle = null;
                /* Determine the order status. */
                var running = item.get("Status");
                /* If the order status is "In progress", change the record style. */
                if (running.value === OrderConfigurationConstants.Order.OrderStatus.Running)
                    item.customStyle = {
                        /* Set the text color to white. */
                        "color": "white",
                        /* Set the background color to green. */
                        "background": "#8ecb60"
                    };
                }
            }
        }
    };
});
```

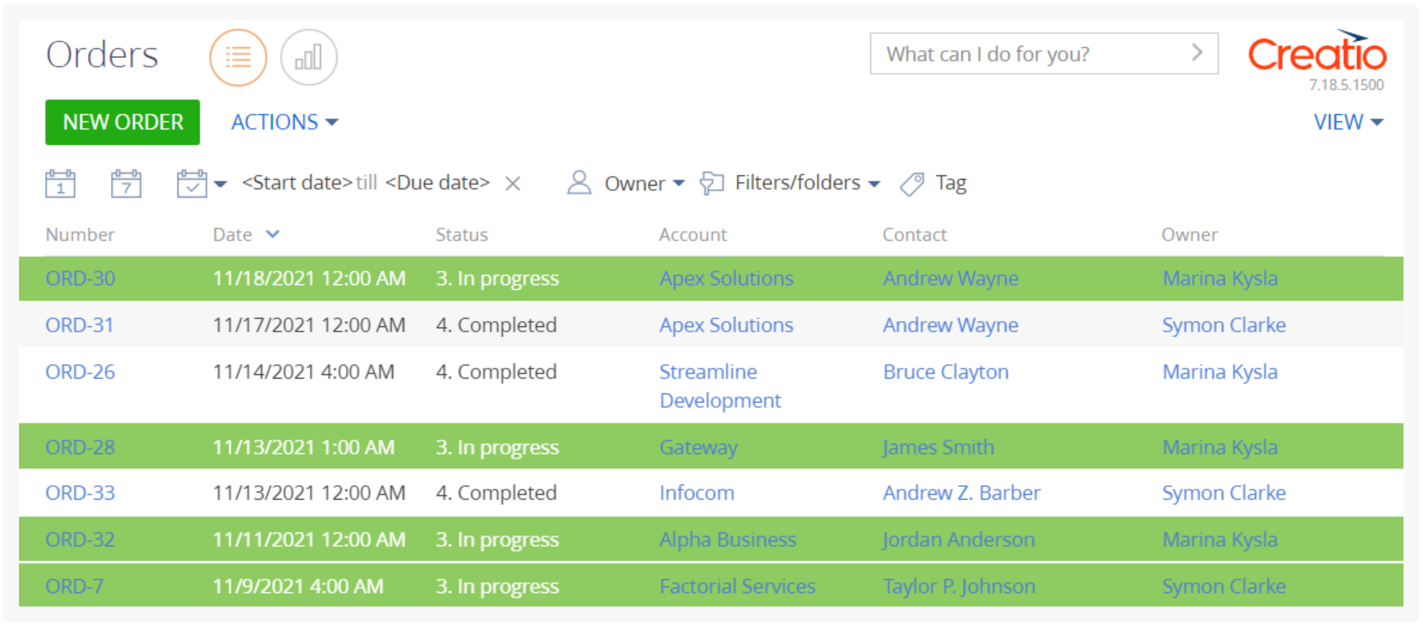
5. Click [*Save*] on the Designer's toolbar.

The outcome of the example

To **view the outcome of the example**:

1. Clear the browser cache.
2. Refresh the [*Orders*] section page.

As a result, Creatio will highlight the orders at the [*In progress*] stage in green.



Number	Date	Status	Account	Contact	Owner
ORD-30	11/18/2021 12:00 AM	3. In progress	Apex Solutions	Andrew Wayne	Marina Kysla
ORD-31	11/17/2021 12:00 AM	4. Completed	Apex Solutions	Andrew Wayne	Symon Clarke
ORD-26	11/14/2021 4:00 AM	4. Completed	Streamline Development	Bruce Clayton	Marina Kysla
ORD-28	11/13/2021 1:00 AM	3. In progress	Gateway	James Smith	Marina Kysla
ORD-33	11/13/2021 12:00 AM	4. Completed	Infocom	Andrew Z. Barber	Symon Clarke
ORD-32	11/11/2021 12:00 AM	3. In progress	Alpha Business	Jordan Anderson	Marina Kysla
ORD-7	11/9/2021 4:00 AM	3. In progress	Factorial Services	Taylor P. Johnson	Symon Clarke

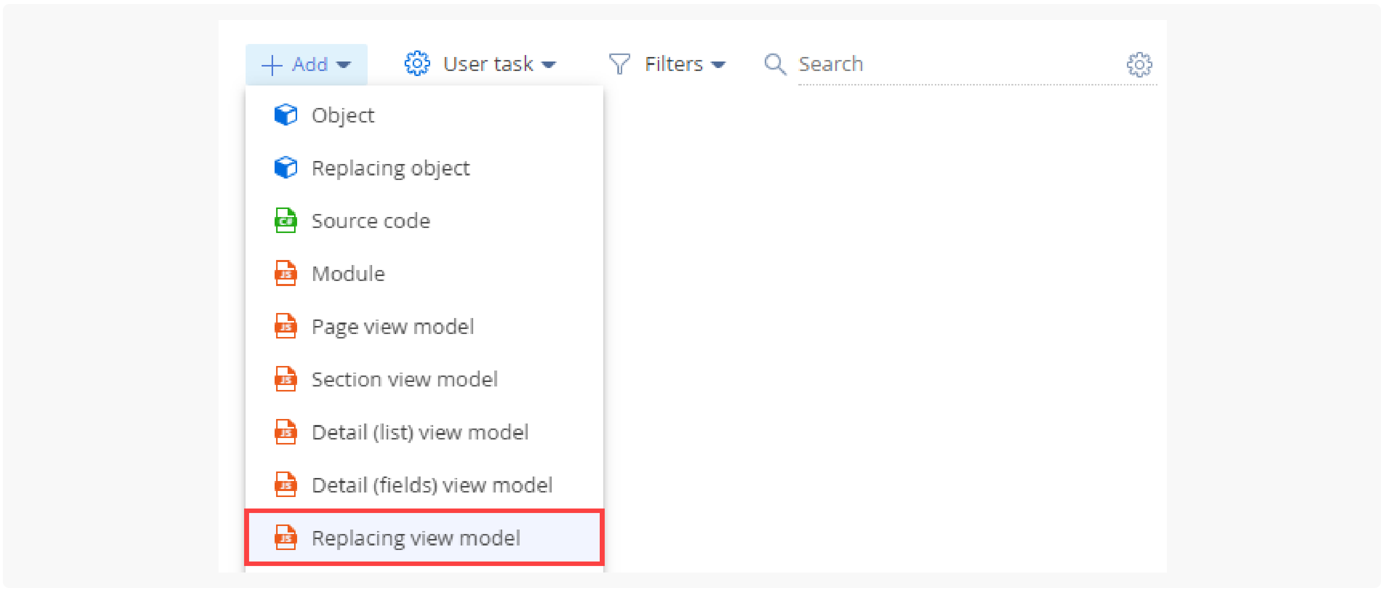
Set up the quick filters section block

Medium

Example. Add a quick filters block to the [*Contracts*] section. Filter contracts by contract start date and owner.

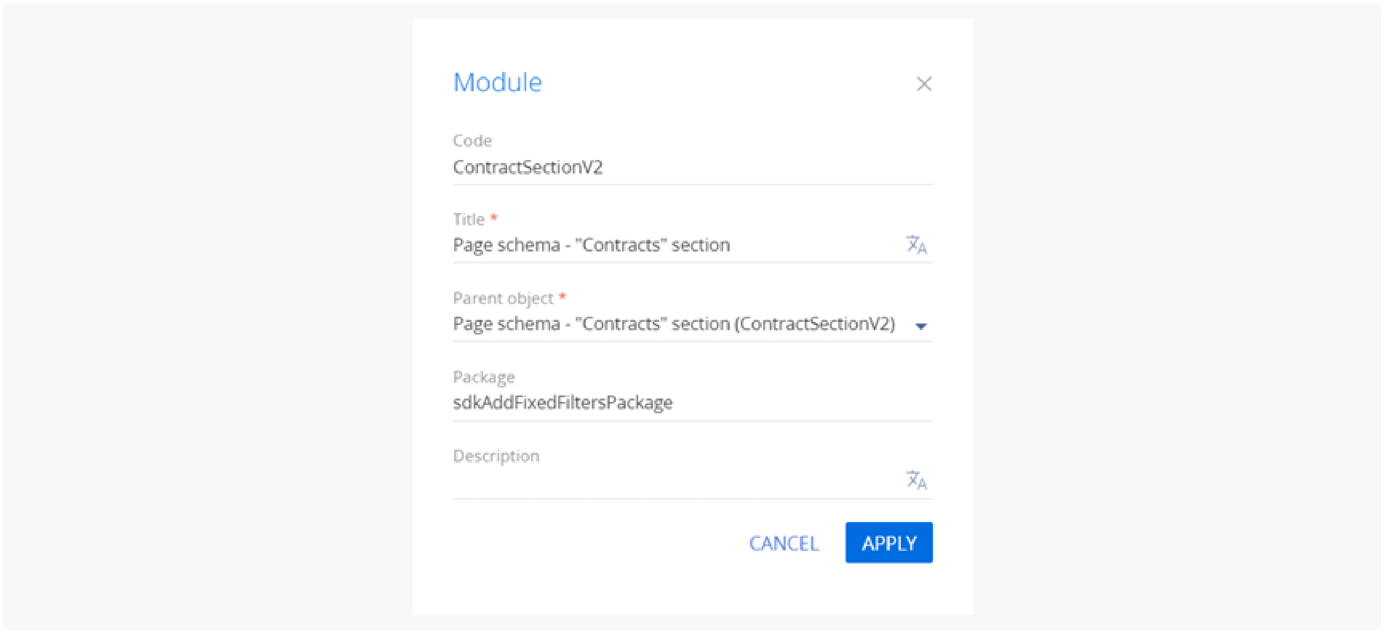
Create a schema of the replacing section view model

1. Go to the [*Configuration*] section and select a custom package to add the schema.
2. Click [*Add*] → [*Replacing view model*] on the section list toolbar.



3. Fill out the **schema properties**.

   - Set [*Code*] to "`ContractSectionV2`".
   - Set [*Title*] to "`Page schema - "Contracts" section`".
   - Set [*Parent object*] to "`ContractSectionV2`".



Module

Code
ContractSectionV2

Title *
Page schema - "Contracts" section

Parent object *
Page schema - "Contracts" section (ContractSectionV2)

Package
sdkAddFixedFiltersPackage

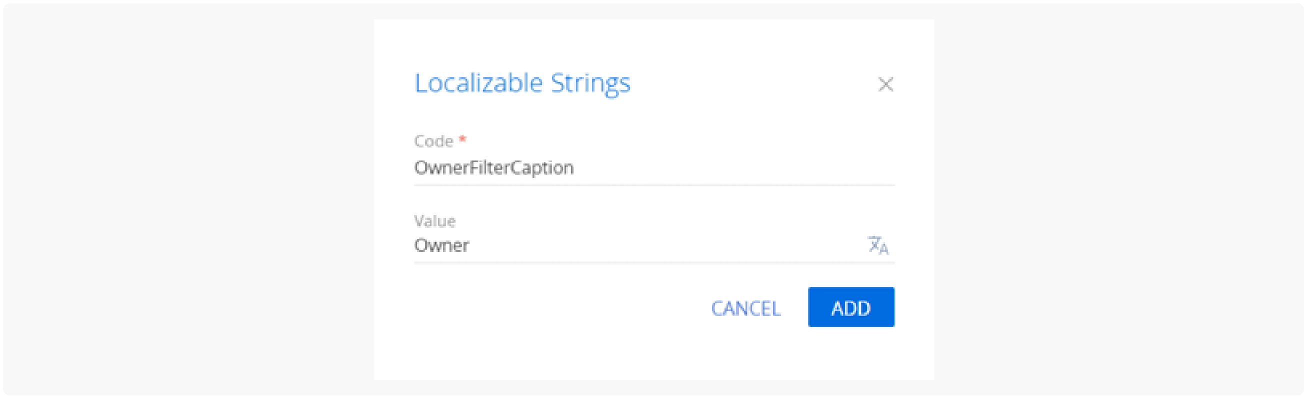
Description

4. Add **localizable strings** that contain filter names.

   a. Add a localizable string that contains the **name of the filter by owner**.

      a. Click the + button in the context menu of the [*Localizable strings*] node.
      b. Fill out the **localizable string properties**.

         - Set [*Code*] to "`OwnerFilterCaption`".
         - Set [*Value*] to "`Owner`".



Localizable Strings

Code *
OwnerFilterCaption

Value
Owner

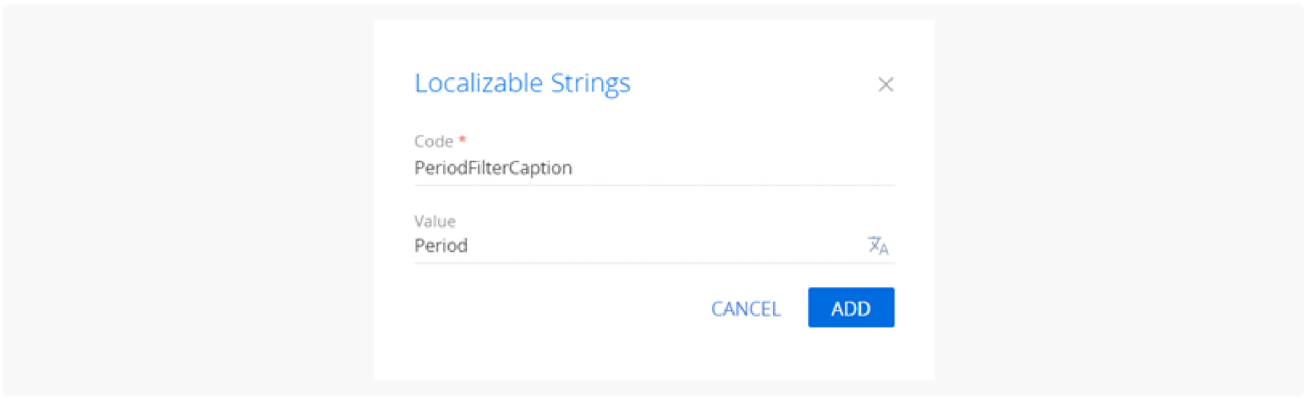
      e. Click [*Add*] to add a localizable string.

   b. Add a localizable string that contains the **name of the filter by period**.

      a. Click the + button in the context menu of the [*Localizable strings*] node.

b. Fill out the **localizable string properties**.

- Set [*Code*] to "`PeriodFilterCaption`".
- Set [*Value*] to "`Period`".



e. Click [*Add*] to add a localizable string.

5. Implement the **filtering behavior**. To do this, implement the `initFixedFiltersConfig()` method in the `methods` property. In the method, create a configuration object with the array of `PeriodFilter` and `OwnerFilter` filters, assign the object link to the `fixedFiltersConfig` attribute of the view model.

   View the source code of the replacing view model schema of the section below.

ContractSectionV2

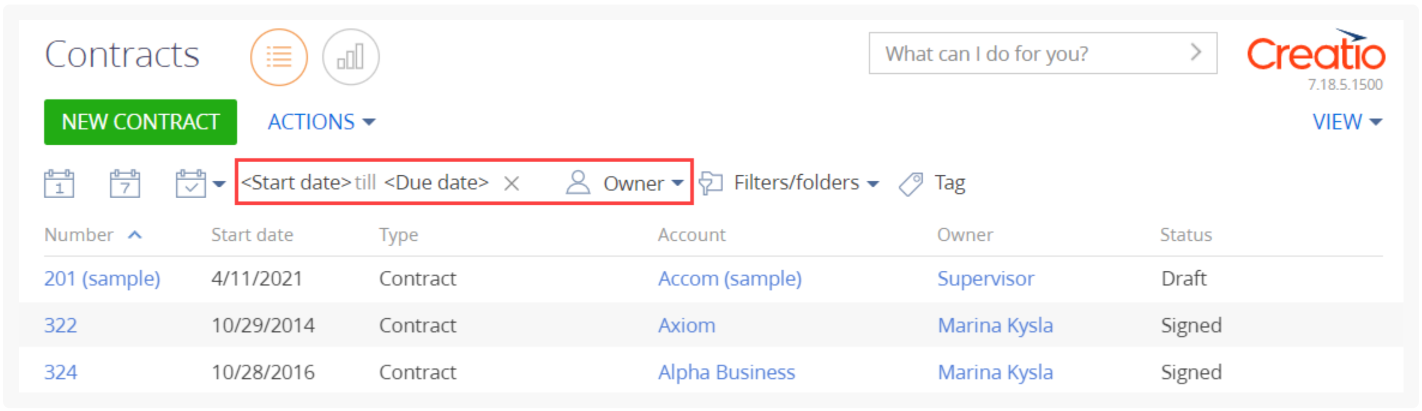
```
define("ContractSectionV2", ["BaseFiltersGenerateModule"], function(BaseFiltersGenerateModule
    return {
        /* The name of the section object schema. */
        entitySchemaName: "Contract",
        /* The methods of the section view model. */
        methods: {
            /* Initialize the fixed filters. */
            initFixedFiltersConfig: function() {
                /* Create a configuration object. */
                var fixedFilterConfig = {
                    /* Specify the section object schema as the object schema for the fixed f
                    entitySchema: this.entitySchema,
                    /* The array of filters. */
                    filters: [
                        /* The period filter. */
                        {
                            /* The filter name. */
                            name: "PeriodFilter",
                            /* The filter caption. */
                            caption: this.get("Resources.Strings.PeriodFilterCaption"),
                            /* Set the data type to data. */
                            dataValueType: this.Terrasoft.DataValueType.DATE,
                            /* The start date of the filtering period. */
                            startDate: {
                                /* Filter data from the [Date] column. */
                                columnName: "StartDate",
                                /* Set the default value to the start of the current week. */
                                defValue: this.Terrasoft.startOfWeek(new Date())
                            },
                            /* Set the end date of the filtering period to the end of the cur
                            dueDate: {
                                columnName: "StartDate",
                                defValue: this.Terrasoft.endOfWeek(new Date())
                            }
                        },
                        /* The owner filter. */
                        {
                            /* The filter name. */
                            name: "Owner",
                            /* The filter caption. */
                            caption: this.get("Resources.Strings.OwnerFilterCaption"),
                            /* Filter data from the [Owner] column. */
                            columnName: "Owner",
                            /* Set the default value to the current user contact specified in
                            defValue: this.Terrasoft.SysValue.CURRENT_USER_CONTACT,
                            /* Set the data type to lookup. */
                            dataValueType: this.Terrasoft.DataValueType.LOOKUP,
                            /* The filter. */
                            filter: BaseFiltersGenerateModule.OwnerFilter
                        }
                    ]
                };
                /* Assign the link to the created configuration object to the [FixedFilterCon
                this.set("FixedFilterConfig", fixedFilterConfig);
            }
        }
    };
});
```

6. Click [*Save*] on the Designer's toolbar.

The outcome of the example

To **view the outcome of the example**, refresh the [*Contracts*] section page.

As a result, Creatio will display the fixed filters block in the [*Contracts*] section. The block will let you filter contracts both by start date and owner.



Number	Start date	Type	Account	Owner	Status
201 (sample)	4/11/2021	Contract	Accom (sample)	Supervisor	Draft
322	10/29/2014	Contract	Axiom	Marina Kysla	Signed
324	10/28/2016	Contract	Alpha Business	Marina Kysla	Signed

BaseSectionV2 schema JS

Beginner

`BaseSectionV2` is the base section schema. Provides the base section logic. Implemented in the `NUI` package. This is a view model schema. Learn more about the schema properties in a separate article: Client schema. All section schemas must inherit from the `BaseSectionV2` schema.

Messages

Base section messages

Name	Mode	Direction	Description
`RenderModule`	Address	Publish	Rerender the message of the dashboard module.
`ReloadDataOnRestore`	Broadcasting	Subscription	Whether to reload data on the following launch.
`SelectedPackageResult`	Address	Subscription	The selected result.

The `Terrasoft.core.enums.MessageMode` enumeration represents the message modes, and the `Terrasoft.core.enums.MessageDirectionType` enumeration represents the message directions. Learn more about the `MessageMode` enumeration in the JS class library. Learn more about the `MessageDirectionType` enumeration in the JS class library.

Attributes

`ChartEditSchemaName` TEXT

The `SchemaName` editing schema.

`IsEmptyChart` BOOLEAN

The flag that marks the schema as empty.

`AnalyticsChartActiveRow` GUID

The active row of the analytics schema

`AnalyticsGridData` COLLECTION

The view collection of the list analytics.

`IsAnalyticsPrintButtonVisible` BOOLEAN

The visibility flag of the button to print out analytical forms.

`AnalyticsData` COLLECTION

The analytics data collection.

`IsAnalyticsActionButtonsContainerVisible` BOOLEAN

The visibility flag of the analytics action buttons.

`AnalyticsDataViewName` TEXT

View the name of the analytics section.

`IsBindDataActionVisible` BOOLEAN

The data binding flag.

The `Terrasoft.core.enums.DataValueType` enumeration represents the attribute data types. Learn more about the `DataValueType` enumeration in the JS class library.

Methods

`checkCanManageAnalytics()`

Whether the user has permission to execute CRUD operations with the schedule. The [*Analytics setup*] (`CanManageAnalytics` code) system operation manages the access permissions.

`onCanManageAnalytics(result)`

Specifies the `CanManageAnalytics` attribute depending on the queried [*Display demo links*] (`ShowDemoLinks` code) system setting and [*Analytics setup*] (`CanManageAnalytics` code) system operation.